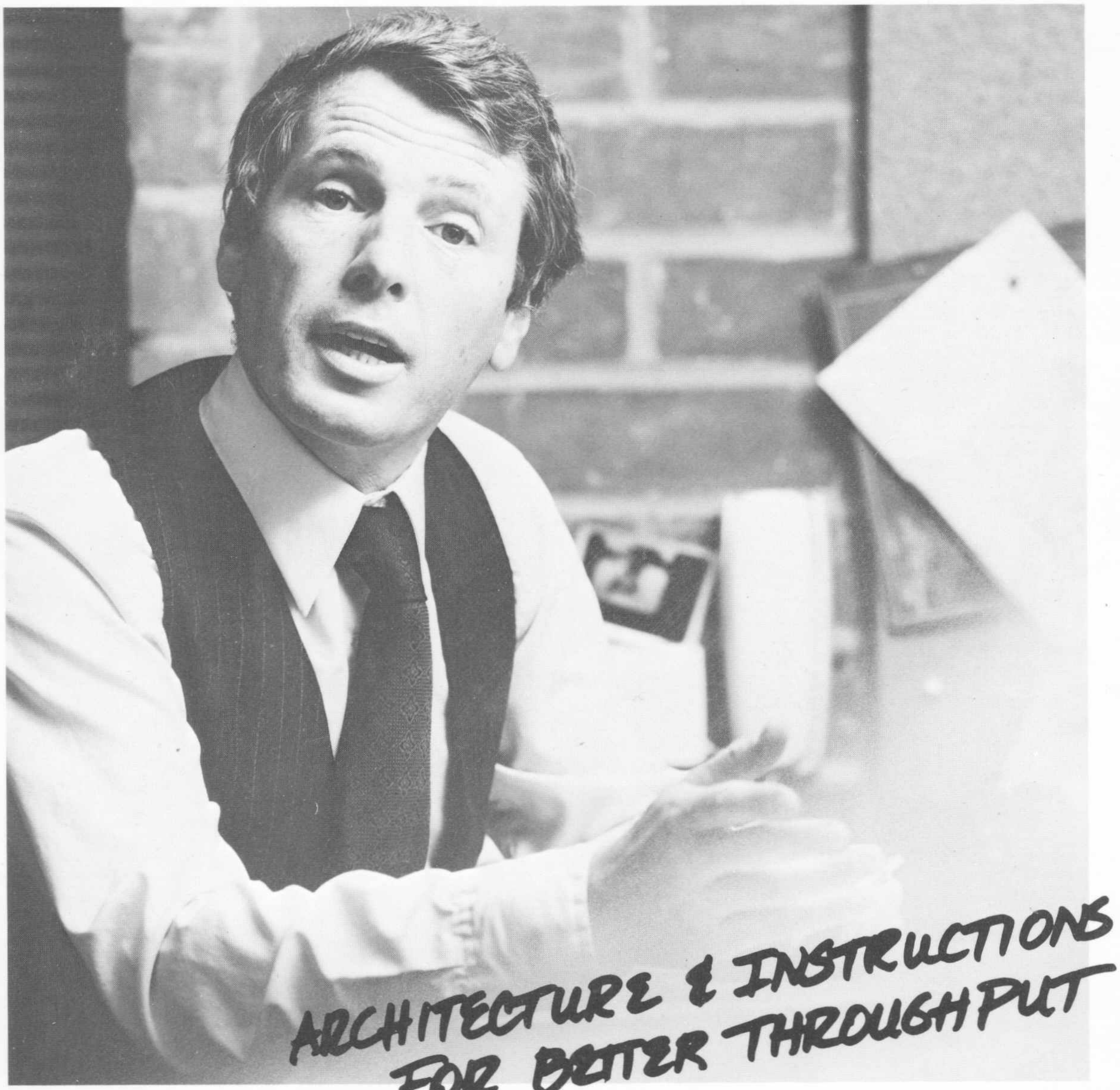
Advanced
Micro
Devices

AmZ8000
User's Manual



ARCHITECTURE & INSTRUCTIONS
FOR BETTER THROUGHPUT

Advanced Micro Devices

AmZ8000 User's Manual

The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating temperature range: 0.1% on MOS RAMs & ROMs; 0.2% on Bipolar Logic & Interface; 0.3% on Linear, LSI Logic & other memories.

INT·STD·123

Advanced Micro Devices reserves the right to make changes in its products without notice in order to improve design or performance characteristics. The company assumes no responsibility for the use of any circuits described herein. Portions of the information contained in this manual were originated by Zilog, Inc. and are used here with their permission.

901 Thompson Place, P.O. Box 453, Sunnyvale, California 94086
(408) 732-2400 TWX: 910-339-9280 TELEX: 34-6306

Printed in U.S.A. 4/81 AMZ-291

CONTENTS

1.0 INTRODUCTION

1.1 HOW TO USE THIS BOOK

This book describes in detail the instruction set of the AmZ8001 and AmZ8002 CPUs. With both the system and application programmer in mind, software aspects of these devices are defined, including the basic architecture of the CPUs and hardware considerations important for a programmer's understanding. Additionally, assembler notations and other information is provided to support programming efforts. More complete details on device hardware or assembler operations can be found in other documents. (See Section 1.3.)

Advanced Micro Devices' School of Advanced Engineering offers courses on the AmZ8000 microprocessor family. Courses include a detailed introduction to the AmZ8000, assembly-level programming on the AmSYS8/8 using the 16-bit real-time emulator, and high-level language programming courses. Other courses cover the Am2900 bit-slice processor family, microprogramming with the AmSYS29, and related topics. Check with your AMD sales office for course outlines and schedules or call toll free (800) 538-8450 extension 3665.

1.2 THE AmZ8000 FAMILY

Advanced Micro Devices has undertaken a significant commitment to the world of 16-bit fixed-instruction-set processors. AMD is bringing to the market:

- A new, advanced processor architecture
- A complete family of LSI peripheral circuits
- A complete family of system support circuits
- A complete family of memories and memory support circuits
- Complete technical documentation
- Effective development system products
- Extensive support software

A large majority of future microprocessor applications will be serviced by a combination of single-chip microcomputer products such as the Am8048 series and by 16-bit microprocessors such as the AmZ8000. Where applications are simple enough, the 8-bit microcomputer chips will tend to be used. Increasing software costs and throughput requirements will cause the 16-bit CPUs to dominate the balance of the designs because they can answer these problems more efficiently. Conventional 8-bit microprocessors will serve a shrinking share of new designs.

In addition to significant increases in throughput that flow directly from the 16-bit structures, improved technology and more sophisticated architectures add even more performance.

Software cost savings are being realized through the use of more powerful instruction sets and sophisticated high-level languages such as PASCAL and C. Language compilers allow programmers to write, debug and document programs more quickly. And saving time is vitally important for such a labor-intensive activity, where costs are rapidly rising. The declining costs of technology-intensive LSI hardware can be used to improve software costs.

AmZ8000 processors, in terms of resources, system features, instructions, interface and architecture represent a major advance in microprocessor sophistication and system-level performance. The processors form the heart of a large family of components, systems, software, documentation and support. In addition to existing peripheral chips, a variety of new, advanced peripherals has been designed to support the AmZ8001 and AmZ8002 processors. Figure 1.2 shows these new MOS/LSI components as well as others that make up the AmZ8000 Family. Full information on these devices and the systems, software, documentation and support available is in The AmZ8000 Family Data Book. (See also Section 1.3.)

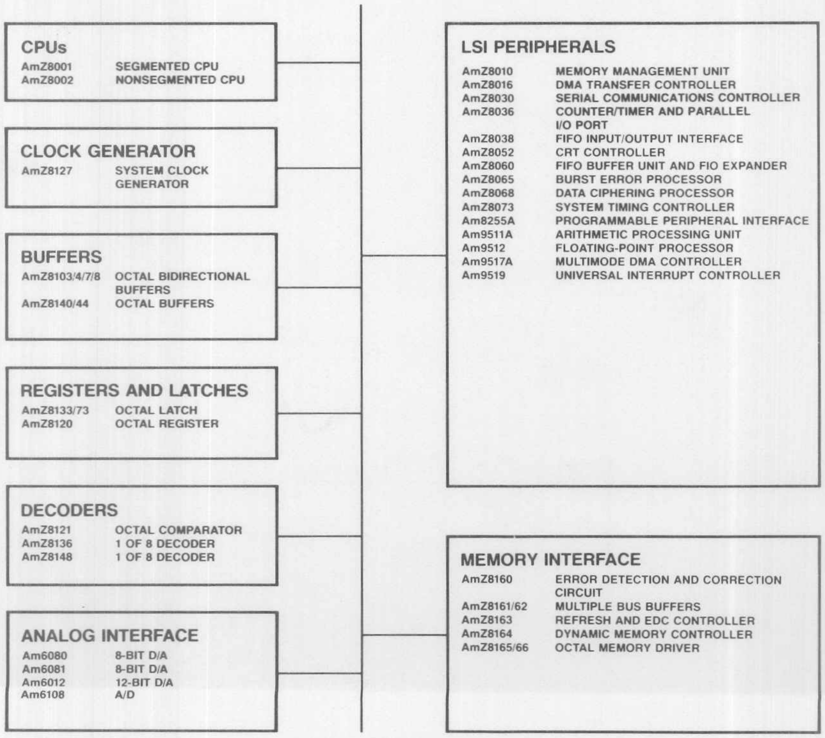

CPUs
AmZ8001 SEGMENTED CPU
AmZ8002 NONSEGMENTED CPU
CLOCK GENERATOR
AmZ8127 SYSTEM CLOCK GENERATOR
BUFFERS
AmZ8103/4/7/8 OCTAL BIDIRECTIONAL BUFFERS
AmZ8140/44 OCTAL BUFFERS
REGISTERS AND LATCHES
AmZ8133/73 OCTAL LATCH
AmZ8120 OCTAL REGISTER
DECODERS
AmZ8121 OCTAL COMPARATOR
AmZ8136 1 OF 8 DECODER
AmZ8148 1 OF 8 DECODER
ANALOG INTERFACE
Am6080 8-BIT D/A
Am6081 8-BIT D/A
Am6012 12-BIT D/A
Am6108 A/D
LSI PERIPHERALS
AmZ8010 MEMORY MANAGEMENT UNIT
AmZ8016 DMA TRANSFER CONTROLLER
AmZ8030 SERIAL COMMUNICATIONS CONTROLLER
AmZ8036 COUNTER/TIMER AND PARALLEL I/O PORT
AmZ8038 FIFO INPUT/OUTPUT INTERFACE
AmZ8052 CRT CONTROLLER
AmZ8060 FIFO BUFFER UNIT AND FIO EXPANDER
AmZ8065 BURST ERROR PROCESSOR
AmZ8068 DATA CIPHERING PROCESSOR
AmZ8073 SYSTEM TIMING CONTROLLER
Am8255A PROGRAMMABLE PERIPHERAL INTERFACE
Am9511A ARITHMETIC PROCESSING UNIT
Am9512 FLOATING-POINT PROCESSOR
Am9517A MULTIMODE DMA CONTROLLER
Am9519 UNIVERSAL INTERRUPT CONTROLLER
MEMORY INTERFACE
AmZ8160 ERROR DETECTION AND CORRECTION CIRCUIT
AmZ8161/62 MULTIPLE BUS BUFFERS
AmZ8163 REFRESH AND EDC CONTROLLER
AmZ8164 DYNAMIC MEMORY CONTROLLER
AmZ8165/66 OCTAL MEMORY DRIVER

Figure 1.2 The AmZ8000 Family

An AmZ8000 Evaluation Board is available from Advanced Micro Computers for quick, hands-on experience with the AmZ8000. Called the Am96/4016, it is a complete small computer with RAM, ROM and several I/O ports. Available software includes a resident monitor and simple line-by-line assembler. The AmSYS™8/8 Microcomputer Development System supports the AmZ8000 Family as well as other microprocessors such as the Am8080A, Am8085A, Z80, and Am8048 Family. The system includes RAM, dual 8-inch floppy disk drives, several serial and parallel interfaces, and a cartridge disk.

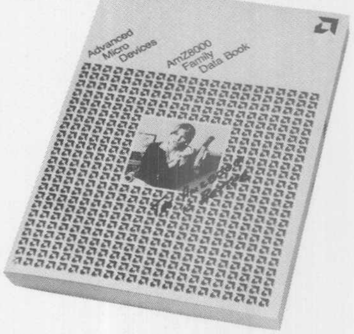
Real-time emulation capabilities are provided by the RTE16 in both stand-alone and AmSYS8/8 plug-in versions. This allows real-time emulation, prototyping, and trace for AmZ8001, AmZ8002 and AmZ8010 MMU designs.

Powerful development software is available with the AmSYS8/8 to make the complex process of product development easier and faster. The software includes a sophisticated disk operating system, macroassemblers, a linking loader, a powerful editor and debugger, and a PASCAL and C compiler with portable library. Additionally, an operating system for the AmZ8000 and cross-software products will be available.

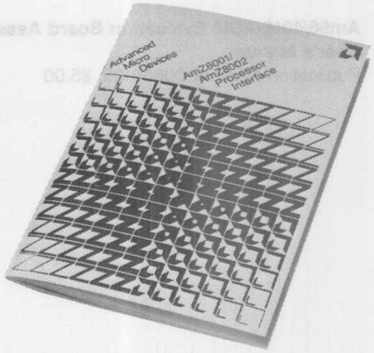
Advanced Micro Devices was conceived on the premise that there is a place in the semiconductor community for a manufacturer dedicated to excellence. This attitude is manifested in many ways throughout the structure of the company and has been maintained throughout the life of AMD. In product assurance procedures, Advanced Micro Devices is unique. Only AMD processes all integrated circuits, commercial as well as military, to the demanding requirements of MIL-STD-883. The AmZ8000 microprocessors and its family of support devices are no exception: every component is 100 percent screened to MIL-STD-883, Method 5004, Class C.

1.3 AmZ8000 LITERATURE

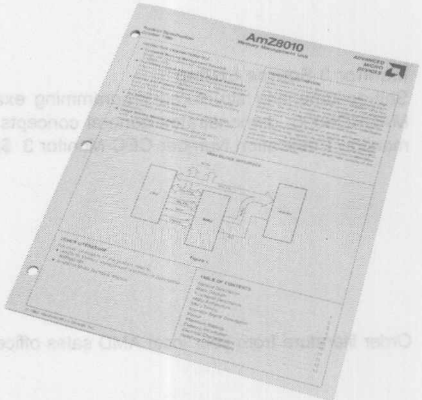
The AmZ8000 Family Data Book
A compilation of data sheets and advanced information on the AmZ8000 CPUs, LSI peripherals, and system support, analog interface, dynamic memory system, and memory components. Also includes development support products: systems, evaluation boards, emulators and software. AmPUB-098



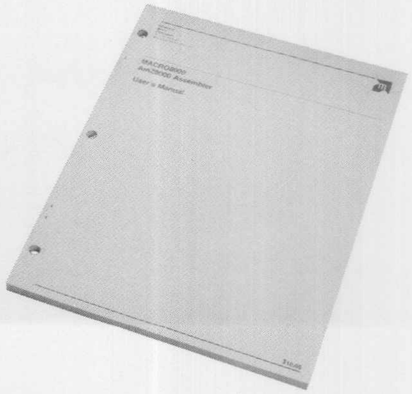
AmZ8001/AmZ8002 Processor Interface
Describes hardware interconnections between CPU and peripherals. Describes interrupt daisy chain and multimicroprocessor systems. AmPUB-089



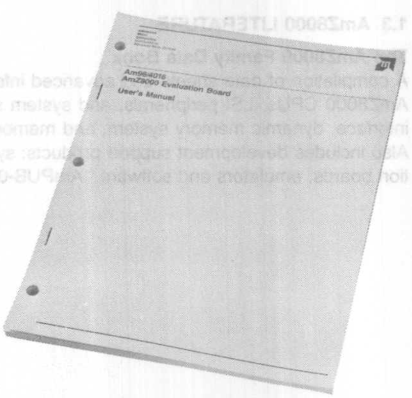
AmZ8010 Memory Management Unit
Complete product specification, including functional description, architecture, and timing. AMZ-192



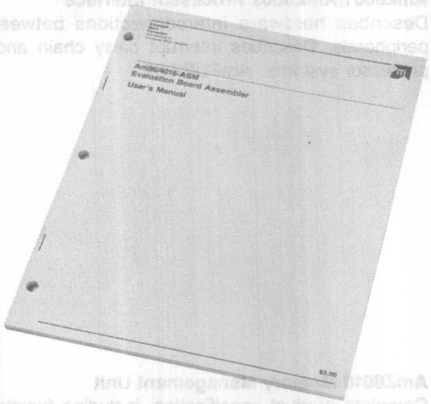
MACRO8000 Assembler User's Manual
Publication Number 00680119 $10.00



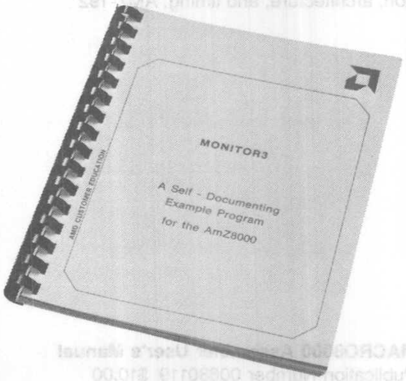
Am96/4016 AmZ8000 Evaluation Board User's Manual
Publication Number 00680131 $10.00



Am96/4016-ASM Evaluation Board Assembler User's Manual
Publication Number 00680138 $5.00



Monitor 3 Source Listing
Self-documenting structured programming example written in MACRO8000; demonstrates general concepts required in any monitor. Publication Number CEC-Monitor 3 $25.00



Order literature from your local AMD sales office.

2.0 CPU ARCHITECTURE

2.1 INTRODUCTION

The AmZ8000 is an advanced 16-bit microprocessor designed to span a wide variety of applications. Its features allow effective use in complex, high-throughput systems, yet it remains efficient for simpler systems, as well. This high-end 16-bit processor and its family of auxiliary devices support advanced systems ranging from simple stand-alone computers to complex parallel processing, multitasking and multiuser systems.

The AmZ8000 CPU represents a major advance in microcomputer architecture by offering many minicomputer and mainframe features in a microprocessor chip. The AmZ8000 CPU is available in two configurations, for use with or without an external memory management device that allows variable segment sizes and implements memory protection and relocation features. The segmented CPU, the AmZ8001, can directly address up to 8 megabytes of memory. The nonsegmented AmZ8002 can directly address 64K bytes.

Abundant CPU resources include numerous registers, many data element types, a large instruction set and eight user-selectable addressing modes. The CPU resources exhibit a consistency and regularity not found in previous microprocessor architectures. Regularity of register organization, of data types, instructions and addressing modes greatly simplifies the programming process and reduces program length.

The AmZ8000 CPU is partitioned into two modes for system (privileged) and normal (nonprivileged) operations. The instruction set similarly is partitioned and includes system-only instructions, such as the I/O instructions, not directly accessible to the normal user. Likewise, stack registers are duplicated in hardware to support both system and normal stacks. Such resources make advanced multiuser and multitasking systems very easy to implement.

The AmZ8000 achieves high throughput with a relatively low clock rate, and can use memories that have a comparatively long access time. The design also includes built-in memory refresh capability with a setable refresh rate that accommodates a variety of dynamic memories.

Compiler, compiler-produced and operating system code all run efficiently on the AmZ8000. The AmZ8000 supports compilers with features such as consistent instruction set, large address space, relocation, multiple stacks and specific instructions (PUSH, POP, INCREMENT and TEST).

Operating systems are supported by features such as system and normal modes, system and normal stack, specific instructions (SYSTEM CALL, LOAD PROGRAM STATUS and privileged instructions), and by a sophisticated interrupt and trap structure. This structure includes three types of interrupts (nonmaskable, nonvectored and vectored) and four types of traps for system calls, privileged instructions, other special instructions, and segmentation.

Multimicroprocessor systems are supported in software by exclusion and synchronization instructions and in hardware by the Micro In input and Micro Out output.

2.2 CPU RESOURCES

Not only must the address space of an advanced architecture be large, but its CPU resources must be abundant enough for the solution of large problems.

The resources of the AmZ8000 CPU can be listed as follows:

- Regular general-purpose register architecture
  - useable as 8-bit, 16-bit, 32-bit, and 64-bit registers
  - user-defined as data, address, index, stack, counter registers
- 8M byte direct addressing range
  - separate code, data, stack spaces
  - 32M byte address space supported conveniently
- Software compatibility between the AmZ8002 and AmZ8001
- System (privileged) and normal (nonprivileged) operating partition
- Powerful instruction set with flexible addressing modes
  - over 110 instruction types
  - eight addressing modes
  - autoindexing instructions
  - string instructions with repeat and nonrepeat versions
- Data types including bits, digits, bytes, words, long words, byte strings, word strings, addresses
- Sophisticated exception processing capabilities
  - nonmaskable, nonvectored, and vectored interrupts
  - four types of traps, including segment trap from memory management unit
  - extended processing interface
- Five types of transfers including memory, I/O, and three daisy chains: interrupt request, bus request, and resource request
- Multimicroprocessing facilities

2.2.1.1 Two CPU Versions

The AmZ8000 CPU is offered in two versions: the AmZ8001 48-pin segmented CPU and the AmZ8002 40-pin nonsegmented CPU. They differ only in the manner and range of memory addressing. Physically, they are the same die with a metal pattern difference determining the CPU configuration.

The AmZ8001 can directly address 8M bytes of memory with its 23 bits of address. The AmZ8002 directly addresses 64K bytes of memory with 16 bits of address. These 16 bits are referred to as the *offset address* as opposed to the additional seven high-order address bits of the AmZ8001. These seven bits define the *segment address* and that is why the AmZ8001 is referred to as the segmented CPU. While the nonsegmented CPU directly addresses one segment of 64K bytes of memory, the segmented CPU can directly address 128 such segments for a total of 8M bytes of memory.

In addition to the seven segment address pins the segment trap pin is available on the 48-pin AmZ8001. This provides an additional interrupt input to the CPU for interfacing to a memory management unit such as the AmZ8010.

In this book AmZ8000 CPU is used to refer to either the AmZ8001 or AmZ8002 CPU. Figures 2.2.1.1 and 2.2.1.2 show the functional and connection pin diagrams for the CPUs.

2.2.1.2 Register Resources

The AmZ8000 offers sixteen 16-bit general-purpose registers in addition to special system registers. All sixteen 16-bit registers may be used as accumulators and all but one can serve as index registers. The first eight of these 16-bit registers may be used as sixteen 8-bit byte registers. They may also be used as sixteen 16-bit word registers, as eight 32-bit long-word registers, or as four 64-bit quadruple-word registers.

The CPU architecture allows the creation and maintenance of stacks in memory. Any of the general-purpose registers (with one exception) can be designated as a stack pointer in the PUSH and

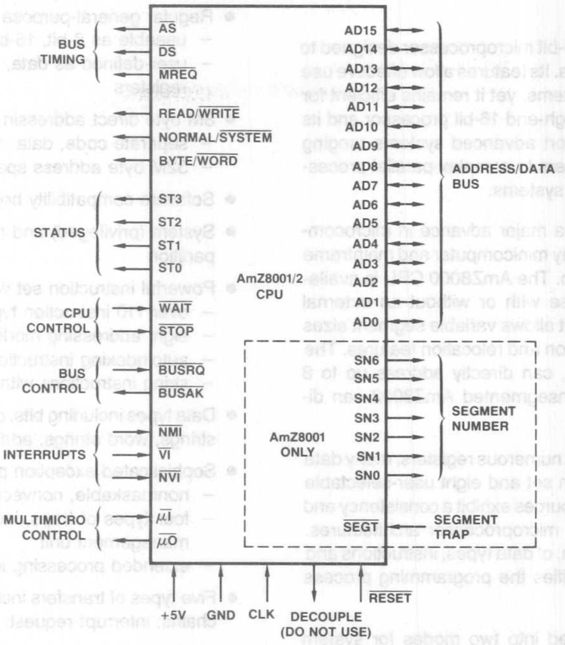
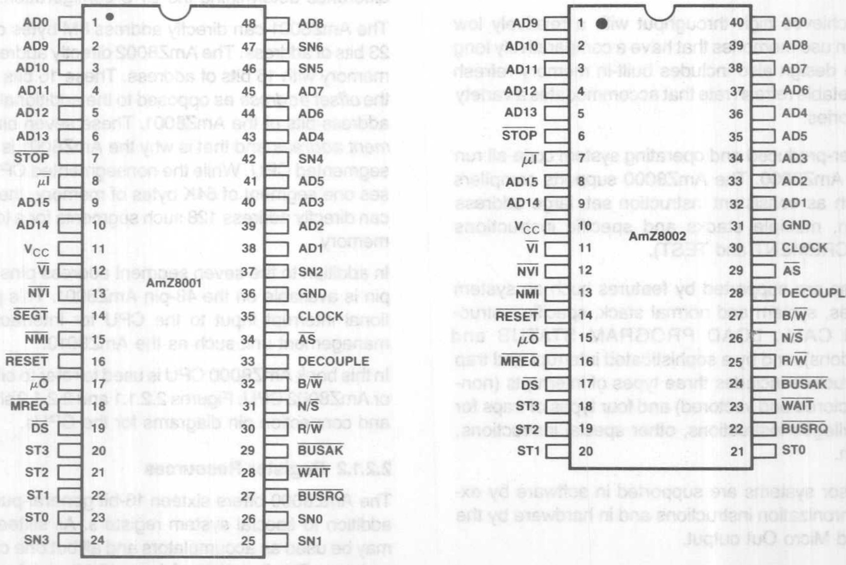


Figure 2.2.1.1 CPU Pin Functions



Note: Pin 1 is marked for orientation.

Figure 2.2.1.2 CPU Connection Diagram – Top View

POP instructions. For the CALL and RETURN instructions specific general-purpose registers are implied as stack pointers as described later.

To support the system and normal modes the specific general-purpose registers implied as stack pointers are duplicated in hardware.

Special registers of the CPU include a program counter, a flag and control word register, a new program status area pointer register (for interrupt or trap context switching), and a refresh counter/register to facilitate dynamic memory system implementation.

2.2.1.3 Instruction Set Resources

The AmZ8000 provides the following groups of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

These instructions are available for both the AmZ8002 nonsegmented CPU and the AmZ8001 segmented CPU.

Over 410 meaningful combinations of instruction types, data elements and addressing modes are available. The AmZ8000 also provides signed-multiply and signed-divide instructions implemented in hardware for both 16-and 32-bit values.

The AmZ8000 supports seven main data types: bits, BCD digits, bytes, words (16 bits), long words (32 bits), byte strings, and word strings. Additionally, many other data elements such as memory addresses, I/O addresses, segment table entries and program status words are also provided.

The eight user-selectable addressing modes include five main modes: Register(R), Indirect Register (IR), Direct Address (DA), Indexed (X) and Immediate (IM). For certain instructions, there are several other addressing modes: Base Address (BA), Base Indexed (BX), Relative Address (RA), Autoincrement and Autodecrement. String instructions such as I/O, translate, and others also have repeat and nonrepeat versions.

Compared to other microprocessors or to 16-bit minicomputers, the number and power of individual instructions has been greatly increased. Over 110 distinct instruction types are available with the AmZ8000 CPUs, compared to approximately 60 for the PDP 11/45. With few exceptions, byte, word and long-word data elements can be processed by all the instructions. Each instruction – again with few exceptions – can use any of the five main addressing modes.

Instruction prefetch (pipelining) has been designed into the CPU to improve overall execution times wherever possible. Also, special opcodes and hardware is designed into the CPU to allow expanding the instruction set externally with closely-coupled extended processing devices.

2.2.1.4 Other Resources

The AmZ8000 CPU outputs status information via four status lines (ST0-ST3) and the System/Normal line ($S/\overline{N}$). These define the operating status of the CPU and can also be used to efficiently extend the addressing range. This is done by allocating physical memory to specific usage.

A very convenient division is to separate physical memory into code, data, system, and normal spaces for either an AmZ8001 or AmZ8002 system. For example, the direct addressing range of 64K bytes of the AmZ8002 can be increased to 256K bytes with such a division.

Other resources and signals are provided for memory and I/O addressing and data transfers, interrupt and trap processing, CPU and bus control, and multimicro control. These are discussed elsewhere.

The AmZ8000 segmented CPU can be run in the nonsegmented mode. This is discussed elsewhere.

2.2.1.5 Multimicroprocessor Support

The AmZ8000 exclusion/serialization mechanism is designed for multimicroprocessor systems. Any CPU in a multimicroprocessor system can exclude all other asynchronous CPUs from any critical shared resource by using the Micro In ($\overline{\mu I}$) input and Micro Out ($\overline{\mu O}$) output in conjunction with the REQUEST, TEST μI, SET μO and RESET μO instructions.

In addition, the large address space of the AmZ8000 is a beneficial feature in multimicroprocessor systems.

2.2.2 Large Addressing Space

High-level languages, sophisticated operating systems, large data bases, large programs and decreasing memory prices are all accelerating the trend toward larger memories. The AmZ8001 can directly address up to eight megabytes of memory per address space. Four convenient separate address spaces exist in the AmZ8000: code and data for both the system mode and the normal mode.

Larger addressing spaces require longer addresses. This increases the size of instructions and requires larger registers (register pairs) when used for addresses. The impact of this is minimized in the AmZ8001 by segmented addressing features, the use of short addresses in many cases, and by the availability of a large number of general-purpose registers. Where the large address space is not required, the AmZ8002 CPU provides 64K bytes of space in each of four spaces and needs only a single 16-bit word for address reference.

AmZ8000 programs can directly access the entire address space. Since eight megabytes are directly addressable, each instruction has a full address mode where at least 23 bits are set aside for the address. However, the AmZ8000 also offers a mode in which the same address can be expressed by 16 bits in many frequent situations where the higher-order offset bits are zeros (short offset mode).

Alternative methods commonly employ fixed internal registers that contain address extensions. Although these methods use shorter addresses, the byte savings are lost because many instructions are required to explicitly manage the contents of the registers. The AmZ8000 can use these methods; however, it also provides direct addressing that removes the necessity for those extra instructions and unburdens the programmer from managing the register contents. It also has no speed loss for short offset addresses and the time lost in the case of long offsets is smaller than the time required to load an internal register.

Another important feature provided to the programmer is the ability to distinguish between system code and normal code, and – in both cases – the additional capability of distinguishing between instruction space, data space and stack space. If this feature is utilized, the AmZ8000 can address up to 48 megabytes of memory.

2.2.3 Segmented Memory Addressing

Segmentation is a powerful and useful technique because it forms an efficient way of dividing an address space into different functional areas. This allows a user to use a different segment for each different area, such as areas for storing procedure instructions, holding global variables, storing multiple user common data or code, or serving as a buffer area for processing large, disk-resident data bases.

The AmZ8001 uses segmented memory addressing not only to address the large amount of memory (8M bytes) but to provide the features required by advanced memory – intensive systems. For applications that do not require a large addressing space, the nonsegmented AmZ8002 CPU is available.

A segmented address is made of two parts: a segment number (7 address bits) and an offset value (16 address bits). The AmZ8001 can designate up to 128 segments each containing up to 64K bytes.

Code written for the nonsegmented AmZ8002 CPU can run in one segment of the segmented AmZ8001 CPU when it is operated in the nonsegmented mode. *Thus, full code compatibility exists between the two versions.*

2.2.4 Memory Management

Complex, large memory systems require not only capabilities of handling larger addressing spaces, but memory management features as well. Variable sized segments, logical to physical relocation, and memory protection can be provided with a separate memory management device. These features will extend the life of the architecture by avoiding memory address limitations that have hampered microprocessors in the past. A memory management system can be employed to provide system support such as segment swapping, memory access monitoring, and memory segment protection of various types (code only, DMA only, etc).

When segmentation is combined with an address translation mechanism to provide relocation capability, the advantages of segmentation are even greater. Then, segments can be of variable user-specified sizes and located anywhere in memory.

To meet such requirements of memory-intensive applications, a memory-management device has been designed to work closely with the AmZ8001 CPU. It is designated the AmZ8010 Memory Management Unit (MMU) and is based on the concept of segmented memory addressing.

The MMU manages the large address space by providing real-time segment relocation from logical to physical address space. It also monitors memory accesses, changes, and other protection attributes (size, CPU inhibit, read only, system only/normal inhibit, execute only/data-stack inhibit, CPU only/DMA inhibit).

This external memory management device essentially doubles the silicon area available to the microprocessor. Hence, it also doubles the logic available to the designer for implementing more features than otherwise would have been possible.

The AmZ8001 is designed to work with or without the MMU. If used, one or more MMUs can be employed for complex memory-intensive systems.

2.2.5 Code Density

Microprocessor speed is largely dependent on the number of executed instruction words. Therefore, code density is an important issue. The AmZ8000 offers several advantages in this respect.

The number of words required to specify frequent instructions (those instructions encountered by the assembler most frequently) has been minimized. This results in one word used for *each* JUMP RELATIVE, CALL RELATIVE, LOAD BYTE REGISTER IMMEDIATE and LOAD WORD REGISTER IMMEDIATE (for small immediate values).

A short offset mechanism is also designed to allow an address to be reduced to a single word. It can be automatically invoked by assemblers and compilers.

Finally, the largest reductions in size and increases in speed result from the consistent and regular structure of the architecture, and the greater power of the instruction set – factors that allow fewer instructions to accomplish a given task. Compared to previous microprocessor designs, AmZ8000 architecture is more regular because its registers, address modes and data element types can be used in a more orderly fashion. Any general-purpose register can be specified to be an accumulator, index register or source register. With few exceptions, all addressing modes can be used with all instructions. Similarly, all the various data types (again with few exceptions) can be used with all the addressing modes.

2.2.6 Throughput

Meaningful evaluations of computer performance will include comparisons based on the execution times of typical programs in typical applications. For the AmZ8000, these applications normally involve high-level language compilers, operating systems and large data-base management. In such areas, the AmZ8000 is five to ten times faster than existing 8-bit microprocessors and two to five times faster than modern 16-bit microprocessors or popular minicomputers such as the PDP 11/34. Furthermore, the AmZ8000 does this with proven N-channel MOS technology and a moderate 4MHz clock rate that allows the use of lower-cost dynamic RAMs. *The AmZ8000 is also available in a 6MHz version*. The AmZ8000 overlaps instruction execution with next instruction fetch and avoids the problems associated with deep unconditional prefetching.

The AmZ8000 can achieve this high degree of performance because its regular architecture does not have critical bottlenecks and because the sophisticated instruction set substantially reduces the number of executed instructions. Some examples of this sophistication are:

- 32-bit operations (including Multiply and Divide) in single instructions
- String manipulation, including Compare and Translate
- Block I/O instructions
- Direct addressing of the entire memory
- Two operating modes (System/Normal or Supervisor/User)
- Powerful interrupt handling

The combination of the powerful instruction set and regular architecture reduces the number of instructions required to execute a program and boosts AmZ8000 performance into the range of well-established minicomputers. (The AmZ8000 is faster than the PDP 11/45 and only slightly slower than the PDP 11/70.)

The following comparisons demonstrate the high execution speed of the AmZ8000 and compare it to the PDP 11/45. Only the 4MHz device is shown in the comparison; times for the 6MHz version would be proportionally shorter. The AmZ8000 is faster in all except Multiply. These tables and graphs show that the advantage of the AmZ8000 is more pronounced when the comparisons involve greater instruction power, more sophisticated addressing modes or longer word lengths. Actual applications programs will take advantage of the more sophisticated aspects of the instruction set and architecture, and will therefore run considerably faster on the AmZ8000 than the PDP 11/45.

TABLE 2.2.6.1 EXECUTION TIME FOR LDB R, src (μs)

Source Addressing Mode	AmZ8000 at 4MHz	PDP 11/45 with 8K
Register	0.75	0.90
Indirect Register	1.75	1.88
Direct Addressing	2.25	2.78
Indexed Addressing	2.50	2.78
Immediate	1.00	1.88

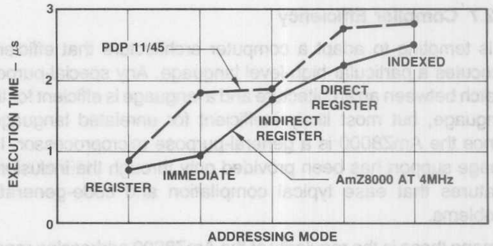
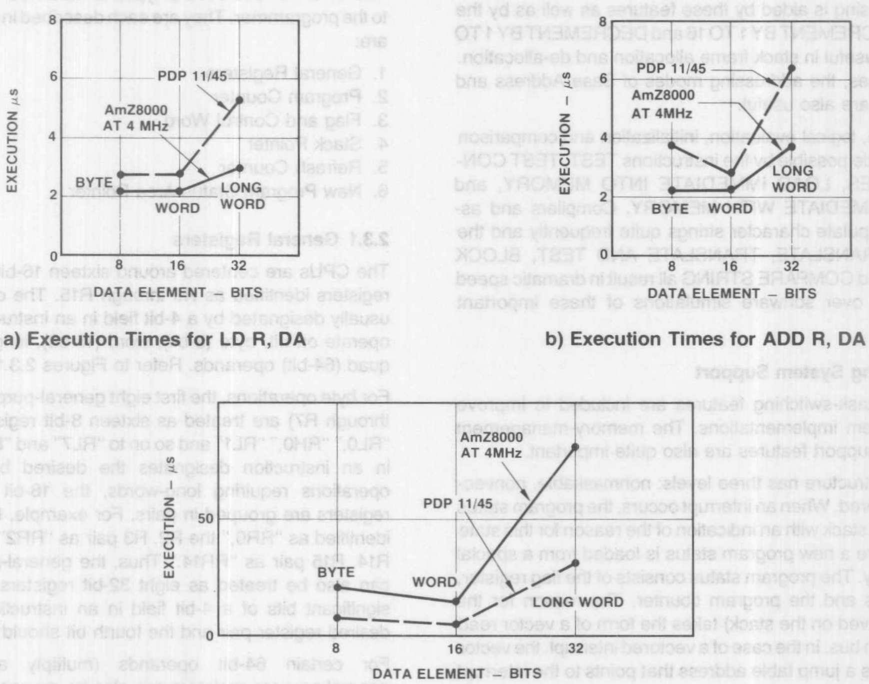


Figure 2.2.6.1 Execution Times LDB R, src for Various Addressing Modes



a) Execution Times for LD R, DA

b) Execution Times for ADD R, DA

c) Execution Times for MULT R, DA

Figure 2.2.6.2

PDP is a registered Trademark of Digital Equipment Corporation

TABLE 2.2.6.2 EXECUTION TIMES FOR LD, ADD AND MULT

		AmZ8000 at 4HMz				PDP 11/45 With 8K		
Operation	Data Type	Inst.	Bytes	Cycles	μs	Inst.	Bytes	μs
LD R, DA	Byte	1	4	9	2.25	1	4	2.78
	Word	1	4	9	2.25	1	4	2.78
	Long Word	1	4	12	3.00	2	8	5.56
ADD R, DA	Byte	1	4	9	2.25	2	6	3.68
	Word	1	4	9	2.25	1	4	2.78
	Long Word	1	4	15	3.75	3	10	6.46
MULT, DA	Byte	3	8	87	21.75	2	6	6.61
	Word	1	4	70	17.50	1	4	5.56
	Long Word	1	4	350	88	17	42	33.94*

*If double floating point is used it is one instruction four bytes and 7.23μs

2.2.7 Compiler Efficiency

It is tempting to adapt a computer architecture that efficiently executes a particular high-level language. Any special-purpose match between an architecture and a language is efficient for that language, but most likely inefficient for unrelated languages. Since the AmZ8000 is a general-purpose microprocessor, language support has been provided only through the inclusion of features that ease typical compilation and code-generation problems.

Among these is the regularity of the AmZ8000 addressing modes and data element types. *Note that any register (except R0) can be used as a stack pointer by its sophisticated PUSH and POP instructions*. Segmentation and relocation are useful features for high-level language procedure implementation. Procedure parameter passing is aided by these features as well as by the instructions INCREMENT BY 1 TO 16 and DECREMENT BY 1 TO 16, which are useful in stack frame allocation and de-allocation. For stack frames, the addressing modes of Base Address and Base Indexed are also useful.

Testing of data, logical evaluation, initialization and comparison of data are made possible by the instructions TEST, TEST CONDITION CODES, LOAD IMMEDIATE INTO MEMORY, and COMPARE IMMEDIATE WITH MEMORY. Compilers and assemblers manipulate character strings quite frequently and the instructions TRANSLATE, TRANSLATE AND TEST, BLOCK COMPARE, and COMPARE STRING all result in dramatic speed improvements over software simulations of these important tasks.

2.2.8 Operating System Support

Interrupt and task-switching features are included to improve operating system implementations. The memory-management and compiler-support features are also quite important.

The interrupt structure has three levels: nonmaskable, nonvectored and vectored. When an interrupt occurs, the program status is saved on the stack with an indication of the reason for this state switching before a new program status is loaded from a special area of memory. The program status consists of the flag register, the control bits and the program counter. The reason for the occurrence (saved on the stack) takes the form of a vector read from the system bus. In the case of a vectored interrupt, the vector also determines a jump table address that points to the interrupt processing routine.

The inclusion of system and normal modes improves operating system organization. In the system mode, all operations are allowed; in the normal mode, certain system instructions are prohibited. The SYSTEM CALL instruction allows a controlled switch of mode, and the implementation of traps enforces these restrictions.

Traps result in the same type of program status saving as interrupts: in both cases, the information saved is pushed on a system stack that keeps the normal stack undisturbed. The LOAD MULTIPLE REGISTER instruction allows the contents of registers to be saved efficiently in memory or on the stack. Running programs can cause program status changes under direct software control with the LOAD PROGRAM STATUS instruction.

Finally, exclusion and serialization can be achieved with the "atomic" TEST AND SET instruction that synchronizes asynchronous cooperating processes.

2.2.9 Software Support

AMD offers a full range of applications support and system support software for the AmZ8000 family. Language compilers, assemblers, utilities, translators for existing programs, etc., are all available to AMD customers. Present users of Z80, Am8080A and Am8085A microprocessors can easily convert their existing software into AmZ8000 code.

Even though there are planned similarities to previous processors, the AmZ8000 represents a major architectural advance over existing designs. There are more instructions, more data types, more addressing modes, larger addressing spaces, and greater instruction complexity. Thus, the full benefits of the AmZ8000 architecture will only be realized by making code changes to existing programs to take advantage of the greater available performance.

2.3 REGISTER ORGANIZATION

The CPU includes several types of hardware registers available to the programmer. They are each described in detail below. They are:

1. General Registers
2. Program Counter
3. Flag and Control Word
4. Stack Pointer
5. Refresh Counter
6. New Program Status Area Pointer

2.3.1 General Registers

The CPUs are centered around sixteen 16-bit general-purpose registers identified as R0 through R15. The desired register is usually designated by a 4-bit field in an instruction. Instructions operate on bit, byte (8-bit), word (16-bit), long word (32-bit) or quad (64-bit) operands. Refer to Figures 2.3.1.1 and 2.3.1.2.

For byte operations, the first eight general-purpose registers (R0 through R7) are treated as sixteen 8-bit registers identified as "RL0," "RH0," "RL1" and so on to "RL7" and "RH7." A 4-bit field in an instruction designates the desired byte register. For operations requiring long-words, the 16-bit general-purpose registers are grouped in pairs. For example, the R0, R1 pair is identified as "RR0," the R2, R3 pair as "RR2" and so on to the R14, R15 pair as "RR14." Thus, the general-purpose registers can also be treated as eight 32-bit registers. The three most significant bits of a 4-bit field in an instruction designate the desired register pair and the fourth bit should be zero.

For certain 64-bit operands (multiply and divide), the general-purpose registers can also be grouped into quads. For example, the R0, R1, R2 and R3 group is identified as "RQ0," the R4, R5, R6 and R7 group as "RQ4" and so on to the R12, R13, R14 and R15 group as "RQ12." The two most significant bits of a 4-bit field in an instruction designate the desired quad register and the remaining two bits should be zero. Table 2.3.1 depicts this organization and gives the four-bit designation used in the source and destination fields of instructions.

The registers may contain operands or address information. When a register pair contains a long-word operand, the even numbered register of the pair holds the most significant 16-bit data while the odd numbered register of the pair holds the least significant 16-bit data. When a register quad is specified for 64-bit data, the first register holds the most significant 16-bits and the last register of the quad holds the least significant 16 bits. For example, R0 is the first register and R3 is the last register of the quad RQ0, R4 is the first and R7 is the last of the quad RQ4 and so on.

In the AmZ8001 (operating in the segmented mode), a register pair will be needed to specify the required 23-bit address. The 7-bit segment number is always specified in the even numbered register and 16-bit offset is specified in the odd numbered register of the pair.

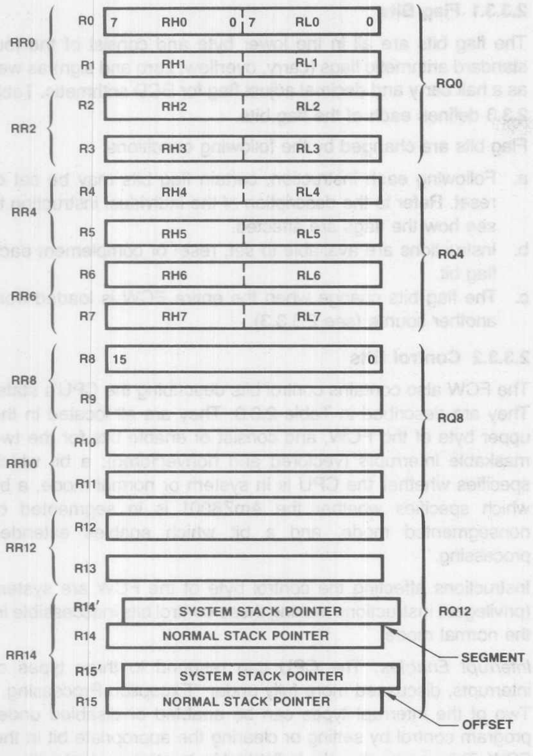


Figure 2.3.1.1 AmZ8001 General Registers

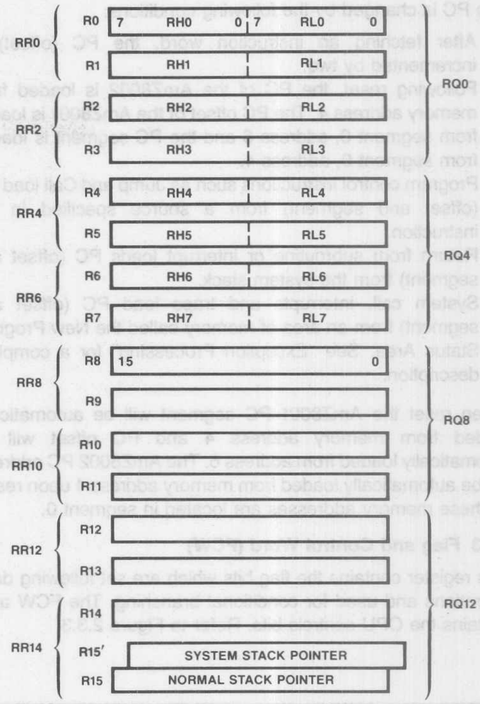


Figure 2.3.1.2 AmZ8002 General Registers

TABLE 2.3.1 GENERAL REGISTER ORGANIZATION AND DESIGNATORS

Register Designator	Byte Mode	Word Mode	Long Word Mode	Quadruple Word Mode
0000	RH0	R0	RR0	RQ0
0001	RH1	R1	–	–
0010	RH2	R2	RR2	–
0011	RH3	R3	–	–
0100	RH4	R4	RR4	RQ4
0101	RH5	R5	–	–
0110	RH6	R6	RR6	–
0111	RH7	R7	–	–
1000	RL0	R8	RR8	RQ8
1001	RL1	R9	–	–
1010	RL2	R10	RR10	–
1011	RL3	R11	–	–
1100	RL4	R12	RR12	RQ12
1101	RL5	R13	–	–
1110	RL6	R14	RR14	–
1111	RL7	R15	–	–

(–Reserved)

All general purpose registers can be used as accumulators. However, R0 in the AmZ8002 (and RR0 in the AmZ8001) cannot be used as an index register or memory pointer. Refer to the section on Address Modes (3.6) and Section 5.2.4.

The highest order general-purpose registers are used as implied stack pointers. For a description of this refer to the section entitled Stack Pointers (2.3.4).

2.3.2 Program Counter

Refer to Figure 2.3.2

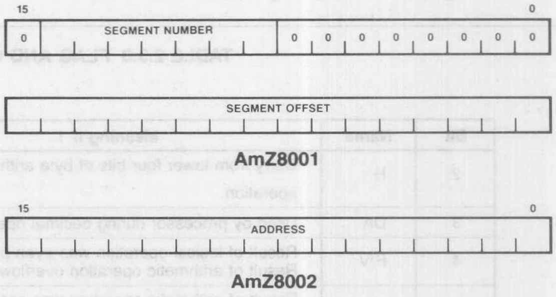


Figure 2.3.2 CPU Program Counters

The program counter points to the address of the next instruction to be fetched from memory. It is a 16-bit register in the AmZ8002 and a register pair in the AmZ8001. The pair contains a 7-bit segment number and a 16-bit offset. The AmZ8000 addresses bytes in memory, instructions occupy full words and are located on even byte addresses (the LSB of the PC is always 0). Following the fetch of each word of an instruction from memory, the PC is incremented by two. In the AmZ8001, only the offset is incremented; the segment number is not changed by incrementing the PC offset.

The PC is changed by the following conditions:

a. After fetching an instruction word, the PC (offset) is incremented by two.

b. Following reset, the PC of the AmZ8002 is loaded from memory address 4. The PC offset of the AmZ8001 is loaded from segment 0, address 6 and the PC segment is loaded from segment 0, address 4.

c. Program control instructions such as Jump and Call load PC (offset and segment) from a source specified in the instruction.

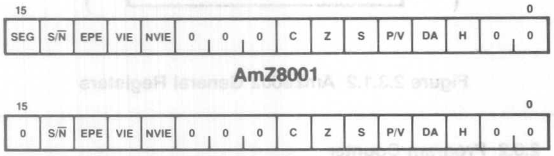
d. Return from subroutine or interrupt loads PC (offset and segment) from the system stack.

e. System call, interrupts and traps load PC (offset and segment) from an area of memory called the New Program Status Area. See "Exception Processing" for a complete description.

When reset the AmZ8001 PC segment will be automatically loaded from memory address 4 and PC offset will be automatically loaded from address 6. The AmZ8002 PC address will be automatically loaded from memory address 4 upon reset. All these memory addresses are located in segment 0.

2.3.3 Flag and Control Word (FCW)

This register contains the flag bits which are set following data operations and used for conditional branching. The FCW also contains the CPU controls bits. Refer to Figure 2.3.3.



AmZ8001

AmZ8002

Figure 2.3.3 CPU Flag and Control Words

2.3.3.1 Flag Bits

The flag bits are all in the lower byte and consist of the four standard arithmetic flags (carry, overflow, zero and sign) as well as a half carry and decimal adjust flag for BCD arithmetic. Table 2.3.3 defines each of the flag bits.

Flag bits are changed by the following conditions:

a. Following each instruction, certain flag bits may be set or reset. Refer to the description of the individual instruction to see how the flags are affected.

b. Instructions are available to set, reset or complement each flag bit.

c. The flag bits change when the entire FCW is loaded from another source (see 2.3.3.3).

2.3.3.2 Control Bits

The FCW also contains control bits describing the CPU's state. They are described in Table 2.3.3. They are all located in the upper byte of the FCW, and consist of enable bits for the two maskable interrupts (vectored and nonvectored); a bit which specifies whether the CPU is in system or normal mode, a bit which specifies whether the AmZ8001 is in segmented or nonsegmented mode, and a bit which enables extended processing.

Instructions affecting the control byte of the FCW are system (privileged) instructions making these control bits inaccessible in the normal mode.

Interrupt Enables: The CPU can respond to three types of interrupts, discussed more fully under "Exception Processing." Two of the interrupt types can be enabled or disabled under program control by setting or clearing the appropriate bit in the FCW. This can be done by individual instructions or by loading a new FCW from another source (see 2.3.3.3).

Extended Processing Enable: The CPU can allow close-coupled extended processing units with the enabling of this control bit. If not set, then upon the execution of any of the special (extended processing) opcodes, a trap will result. See "Extended Processing" for a description of this.

TABLE 2.3.3 FLAG AND CONTROL WORD BITS (FCW)

Bit	Name	Meaning if 1	Meaning if 0
2	H	Carry from lower four bits of byte arithmetic operation.	No carry.
3	DA	Used by processor during decimal operation.	
4	P/V	Result of logical operation was even parity. Result of arithmetic operation overflowed.	Result of logical operation was odd parity. Result of arithmetic operation did not overflow.
5	S	Result of arithmetic operation was negative (MSB = 1).	Result of arithmetic operation was zero or positive (MSB = 0).
6	Z	Result of arithmetic or logical operation was all zeroes.	Result of arithmetic or logical operation contains at least a single one.
7	C	Carry occurred from MSB (sign) bit of arithmetic operation.	No carry.
11	NVIE	Nonvectored interrupts enabled.	Nonvectored interrupts disabled.
12	VIE	Vectored interrupts enabled.	Vectored interrupts disabled.
13	EPE	Enable extended processing.	Implements trap if special extended processing opcode is executed.
14	S/N	Processor in system mode.	Processor in normal mode.
15	SEG	AmZ8001 operating in segmented mode.	AmZ8001 emulating AmZ8002 (nonsegmented).

System/Normal Mode: The CPU can operate in either of two modes; called "system" and "normal" (or "user"). The mode is specified by a bit in the FCW. The differences between the two modes are:

a. Certain instructions cannot be executed in normal mode. These include those instructions which are generally used by an operating system. They are:

- All I/O instructions
- Instructions affecting the control bits (upper byte of FCW)
- Instructions affecting the multiprocessor facility

An attempt to execute one of these instructions results in a trap called "privileged instruction trap." See "Exception Processing" for a detailed description of what happens.

b. A different register is used for R15 (R14 and R15 in the AmZ8001). This general register is intended for use as the primary stack pointer. While any general register except R0 can be used as a stack pointer, certain instructions automatically invoke R15 as the stack pointer for saving system status. The CPU uses a different hardware register (or register pair) for R15 when in the system mode than in the normal mode as registers are duplicated in hardware for both system and normal stack pointer.

The extra register is called R15′ to distinguish it from the normal mode R15. In the AmZ8002, only R15 is separated. In the AmZ8001, both R14 and R15 are separated, since the pair is needed to hold both segment and offset of the stack address. A reference to R15 (or R14) when the processor is in system mode actually refers to R15′ (or R14′). The same reference in normal mode refers to the normal R15 (or R14). All other registers are the same in either mode.

The System/Normal bit is changed only by loading a new FCW (see 2.3.3.3).

Nonsegmented Mode of the AmZ8001. In the AmZ8002, addresses are completely contained in a single 16-bit word. The AmZ8001 uses a segment and offset, requiring two words. All the instructions are identical; only the address references are different. The AmZ8001 can be switched to a nonsegmented mode to execute software assembled for the AmZ8002. This switch is accomplished by the segment bit (15) in the flag and control word.

The segment bit is available only on the AmZ8001. It is always zero on the AmZ8002. It can be changed on the AmZ8001 only by loading a new FCW (see 2.3.3.3). Refer to "Segmented and Nonsegmented Modes" for a description of this feature.

2.3.3.3 Loading a New FCW

The entire Flag and Control Word can be loaded in the following ways:

a. An instruction is available to load it from any source.
b. On an interrupt or trap it is loaded automatically from the New Program Status Area (see "Exception Processing").
c. On a return from an interrupt or trap it is restored to its previous state by a reloading from the system stack.
d. Following RESET, the FCW is loaded from memory location 2 (segment 0 of AmZ8001).

2.3.3.4 Processor Status Information

The contents of the program counter and flag and control word are collectively called the Processor Status Information. When an interrupt or trap occurs, current processor status information is saved and new processor status information is loaded from the New Program Status Area of Memory (see "Exception Processing").

Figure 2.3.3.4 illustrates how the program status groups of the AmZ8001 and AmZ8002 differ. In the nonsegmented CPU the program status group consists of two words: the PC and FCW. In the segmented CPU the program status group consists of four words: a two-word PC, the FCW, and an unused word reserved for future use.

With the exception of the segment enable bit in the AmZ8001 program status group, the flags and control bits are the same for both CPUs.

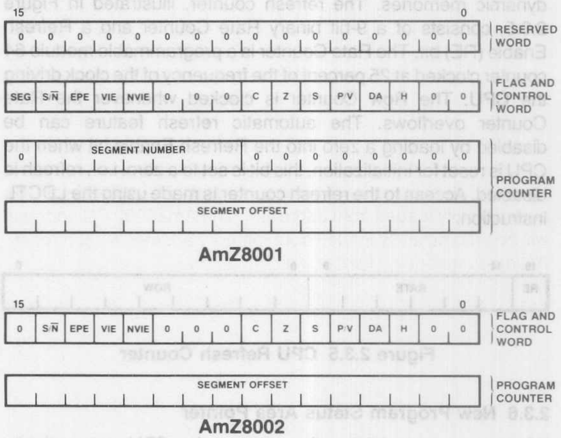


Figure 2.3.3.4 CPU Processor Status Information

2.3.4 Stack Pointers

The architecture allows the creation and maintenace of stacks in the memory. Any of the general-purpose registers (except RR0 in AmZ8001 and R0 in AmZ8002) can be designated as stack pointers in the PUSH and POP instructions. However, for the CALL and RETURN instructions, specific general-purpose registers are implied as stack pointers.

In the AmZ8001 the general-purpose register pair RR14 is the implied stack pointer. The 7-bit segment number is contained in R14 and the 16-bit offset value is contained in R15. The segment number and the offset value form a 23-bit segmented address. For the AmZ8002 the general-purpose register R15 is the implied stack pointer and contains the required 16-bit address. It should be remembered that the implied stack pointers are still general-purpose registers. In other words, certain implied general-purpose registers are given stack pointer attributes in addition to their normal general-purpose characteristics.

The processors can operate in one of two selectable modes: System and Normal. The System mode is sometimes called "supervisor" or "privileged" mode and the Normal mode is sometimes known as "program" or "nonprivileged" mode. Separation of system and normal stacks is desirable in order to facilitate sophisticated system designs. This is accomplished by providing System Stack Pointer in addition to Normal Stack Pointer by duplicating these registers in hardware.

In the AmZ8001 two additional registers, R14′ and R15′, are provided corresponding to R14 and R15. When the AmZ8001 is operating in the system mode, R14′ will be used as the general-purpose register whenever R14 is specified. Similarly R15′ will be used instead of R15 in the system mode for both AmZ8001 and AmZ8002. Thus, the register pair R14′, R15′ (identified as RR14′) is the implied System Stack Pointer for the AmZ8001 and R15′ is the implied System Stack Pointer for the

AmZ8002. Although R14 and R15 are not used in the system mode, instructions are provided such that these two general-purpose registers can be accessed without actually switching the operating mode. The System Stack Pointer will be used during program interruptions to save the pre-interrupt status irrespective of the selected operating mode. Refer to Figures 2.3.1.1 and 2.3.1.2 for a look at these.

2.3.5 Refresh Counter

Both AmZ8001 and AmZ8002 contain a refresh counter for dynamic memories. The refresh counter, illustrated in Figure 2.3.5, consists of a 9-bit binary Rate Counter and a Refresh Enable (RE) bit. The Rate Counter is a programmable modulo 64 counter clocked at 25 percent of the frequency of the clock driving the CPU. The Row Counter is clocked whenever the Rate Counter overflows. The automatic refresh feature can be disabled by loading a zero into the Refresh Enable bit when the CPU is reset for initialization, this bit is set to a zero, i.e., refresh is disabled. Access to the refresh counter is made using the LDCTL instruction.

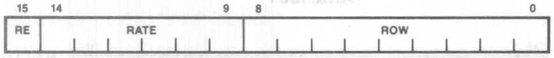


Figure 2.3.5 CPU Refresh Counter

2.3.6 New Program Status Area Pointer

When a program interruption occurs, the CPU automatically saves the program status in the system stack. New program status will be loaded into the FCW and PC. This new program status is obtained from a location in the memory called New Program Status Area. The user defines the begining location of this area by setting the New Program Status Area Pointer(NPSAP). The NPSAP may point to a 256-byte boundary anywhere into memory. The NPSAP is shown in Figure 2.3.6. In the AmZ8001 it consists of two 8-bit registers, one for the segment and one for the most significant eight bits of the offset. On the other hand, only one 8-bit register is used in the AmZ8002. This register specifies the most significant 8-bits of the 16-bit address. The NPSAP is changed by using the LDCTL instruction. See "Exception Processing" for details of the New Program Status Area.

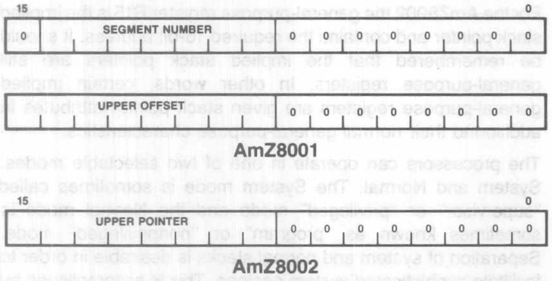


Figure 2.3.6 New Program Status Area Pointer

2.4 EXCEPTION PROCESSING

Program interruptions are divided into two groups – interrupts and traps. In general, an interrupt is an external asynchronous event needing the CPU's attention. A trap usually is a synchronous event resulting from the execution of certain instructions under some specified condition. Also an interrupt (except for the nonmaskable type) may be disabled in the CPU by an appropriate control bit in the FCW; traps cannot be disabled. Procedures followed by the CPU are essentially the same for interrupts and traps.

The descending order of priority for traps and interrupts is: internal traps, nonmaskable interrupt, segmentation trap, vectored interrupt, and nonvectored interrupt. These types are defined below.

When an interrupt or trap occurs, the CPU automatically switches to the system mode and saves the program status information (PC and FCW) plus identifier word on the system stack. The identifier supplies the reason for the interruption and is returned by the interrupt or trap. For external traps (segmentation error) and interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle. Refer also to Figure 5.5.6.

2.4.1 Type of Interrupts

There are three types of interrupts serviced by the CPU. In order of decreasing priority they are: nonmaskable, vectored and nonvectored.

The *nonmaskable interrupt* is always active; it cannot be disabled under program control. It is typically used for power-failure sensing. The nonmaskable interrupt results in a jump to a user defined location in memory. The interrupting device may pass a 16-bit identifier containing information about the interrupt to the program.

The *vectored interrupt* may be masked by clearing bit 12 in the FCW register. If bit 12 is set, then an interrupt causes a jump in memory to an address defined indirectly by the vector supplied by the interrupting device. The device generating the interrupt may pass an 8-bit identifier to the program, along with the 8-bit vector.

The *nonvectored interrupt* may be masked by clearing bit 11 in the FCW register. If bit 11 set, then an interrupt causes a jump in memory to an address defined by the user. A nonvectored interrupt always results in a branch to the same location. The interrupting device may pass a 16-bit identifier to the program.

2.4.2 Traps

There are four types of traps in the CPU: system call instruction, special opcode, privileged instruction in normal mode, and segmentation error. They cannot be disabled.

The identifier in the case of traps (except segmentation error) is the first word of the instruction that caused the trap. This word always contains the instruction opcode.

System Call: The system call instruction is a software trap. This instruction contains an 8-bit field which can be used by the programmer to pass information to the system. The system call results in a branch to a memory address specified by the user.

Special Opcode: There are six special opcodes in the AmZ8001 and AmZ8002 which allow for extended processing or results in a special opcode trap (see "Extended Processing"). An attempt to execute one of them (if the EPE control bit is not set) results in this trap occurring. The program branches to a memory address specified by the user. The opcode causing the trap is passed to the program as an identifier. The six special opcodes, in hex, are: OExx, OFxx, 4Exx, 4Fxx, 8Exx, 8Fxx.

Privileged Instruction: This trap results when an attempt is made to execute one of the system-only instructions when the CPU is in the normal mode. The program branches to a user-defined address and the offending opcode is passed as an identifier. The privileged instructions are: all I/O instructions, instructions which inspect or modify the control bits of the FCW, and those that are involved·in the multimicro (multiprocessor) facility.

Segmentation Error: This trap is used by the AmZ8001 to respond to an addressing error, generally an offset which is outside the defined boundary for a segment or some other segment addressing violation detected by the Memory Management Unit (AmZ8010). The identifier is supplied by the AmZ8010 and is discussed fully in the MMU documentation.

2.4.3 New Program Status Area

After the old program status information (PC and FCW) is stored in the system stack following an interrupt or trap, the new program status information is automatically loaded into the PC and FCW from a specified area in memory. This area is called the New Program Status Area (NPSA) and it contains information for context switches due to each type of interrupt and trap.

The CPU allows the location of the NPSA to be anywhere in the addressable memory space, although it must be aligned to a 256 byte boundary. The New Program Status Area Pointer (NPSAP) register specifies the begining address of the NPSA (see 2.3.6). The CPU uses this address along with an offset, depending upon which type of interrupt or trap occurs, to index into the NPSA table. Table 2.4.3 defines the location of the data in the NPSA.

The NPSAP is a one-word register in the AmZ8002, the lowest byte of which is zero. In the AmZ8001 the NPSAP is stored in a two-word register; the first contains the segment number and the second contains the offset. Again, the lowest byte is zero. The NPSAP is loaded and read using the LDCTL instruction.

(The NPSA must be located in code memory space if separate memory spaces for code and data are being used or when using the code attribute with the Memory Management Unit. This is because of the Status Line decode for a code reference; refer to the MMU documentation.)

The NPSA contains a table of the new flag and control words and starting address (PCs) of service routines for each type of interrupt and trap. The table format is shown in Figures 2.4.3.1 and 2.4.3.2 for the AmZ8001 and AmZ8002, respectively.

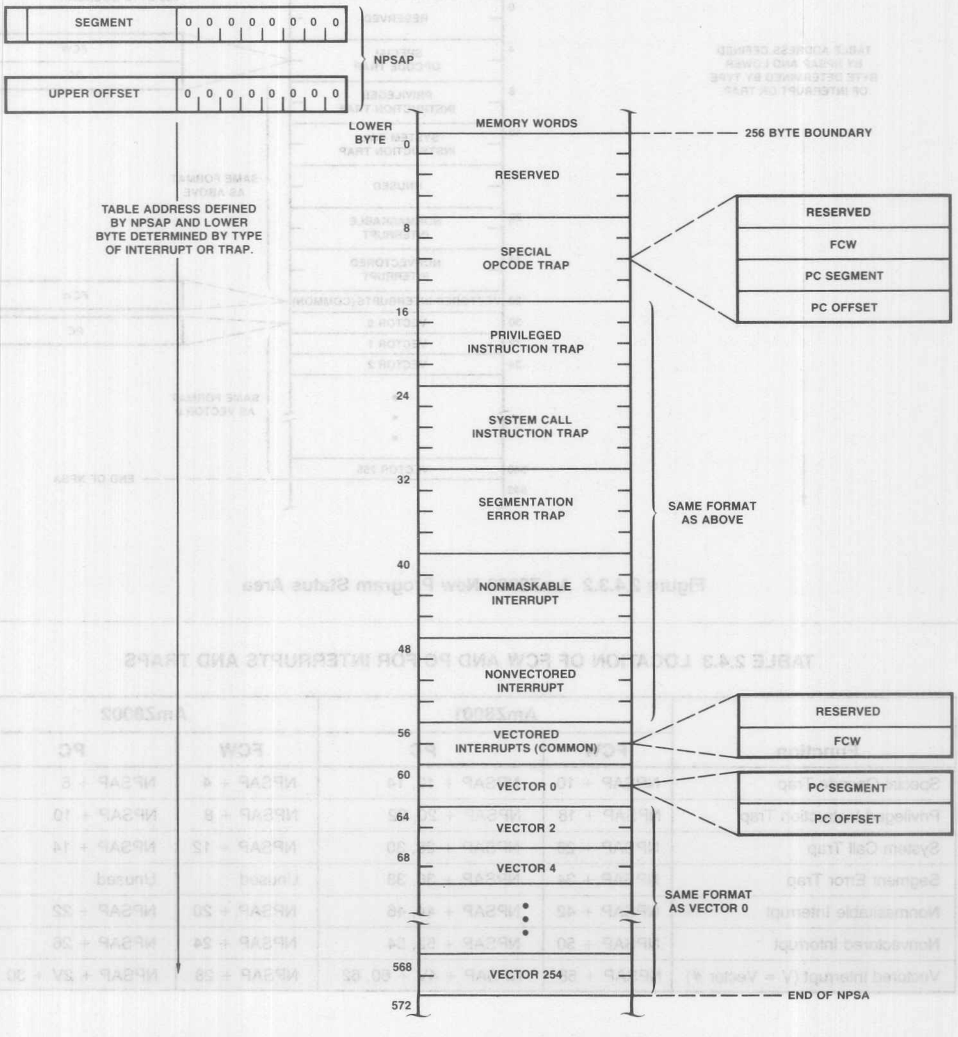


Figure 2.4.3.1 AmZ8001 New Program Status Area

For the AmZ8001 consecutive locations contain 1) reserve word, 2) new FCW, 3) new PC segment and 4) new PC offset for all traps and interrupts except vectored interrupts. The vectored interrupts share a common reserved word and FCW, and each vector has its own PC segment and offset. For the AmZ8002 there are just two words for each entry, the FCW and PC. The vectored interrupts share a common FCW and have separate PCs. Refer to Section 2.4.5.

2.4.4 Exception Processing Sequence

When an interrupt or trap occurs, the current program status information is pushed onto the system stack in the following order: program counter (or PC offset, then segment in the AmZ8001), contents of FCW register, Identifier. This is illustrated in Figure 2.4.4.1. (See also Figure 5.5.6.) The identifier is the data defined above for the various types of interrupts and traps. The service routine can access the identifier by simply addressing the word pointed to by the system stack pointer since it is at the top of the stack.

The program counter and flag control word are reloaded with values located in memory in the New Program Status Area. The new program status will be loaded from an address whose upper 8 bits (and segment number in the AmZ8001) are stored in the NPSAP and whose lower 8 bits are determined by the particular trap, interrupt or vector occurring. The CPU, of course, then begins execution immediately at the address found in the New Program Status Area. This address can also be anywhere in memory. The entire process is illustrated in Figure 2.4.4.2.

To service an interrupt or trap requires 34 CPU clocks.

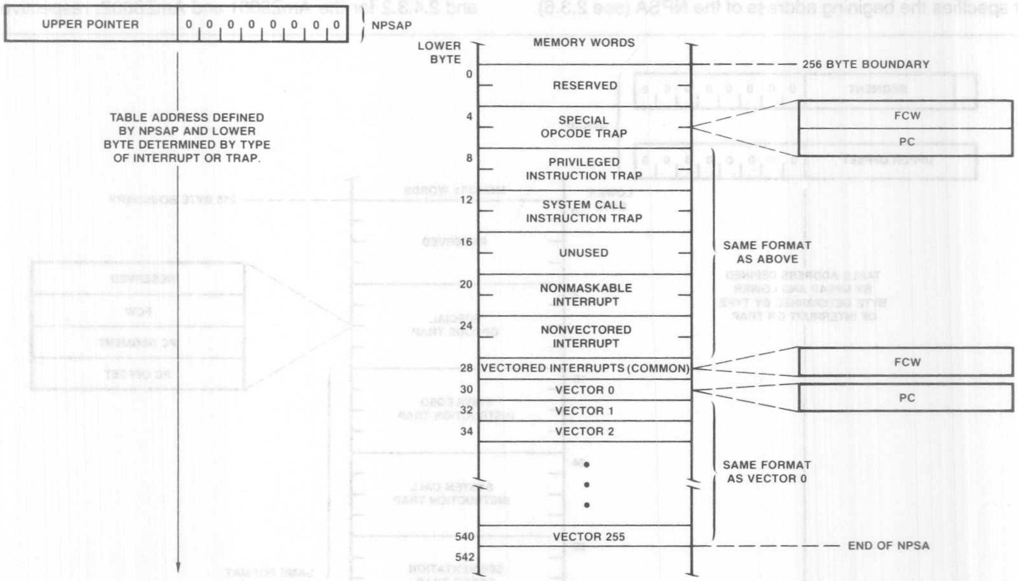
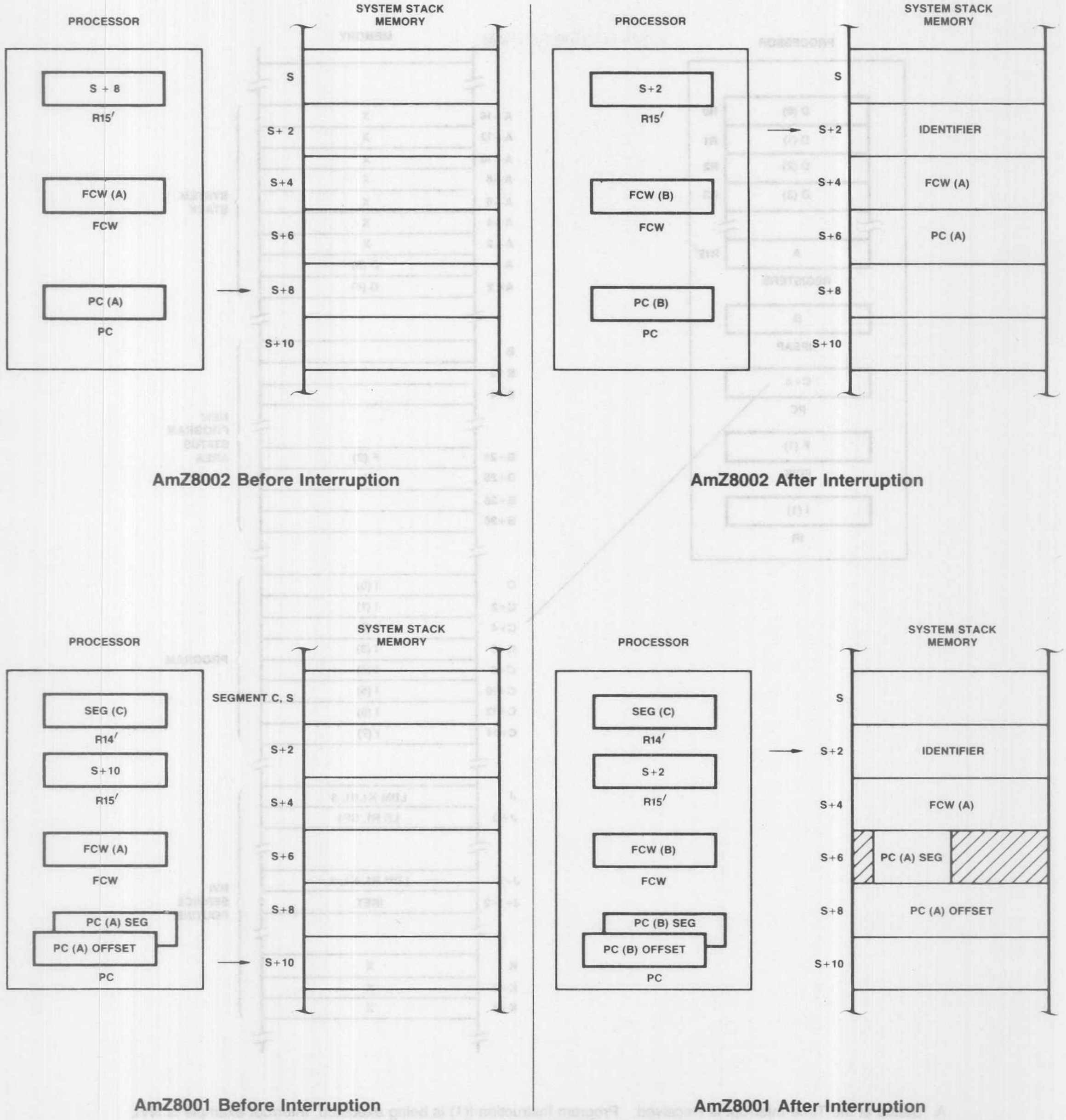


Figure 2.4.3.2 AmZ8002 New Program Status Area

TABLE 2.4.3 LOCATION OF FCW AND PC FOR INTERRUPTS AND TRAPS

Function	AmZ8001		AmZ8002	
	FCW	PC	FCW	PC
Special Opcode Trap	NPSAP + 10	NPSAP + 12, 14	NPSAP + 4	NPSAP + 6
Privileged Instruction Trap	NPSAP + 18	NPSAP + 20, 22	NPSAP + 8	NPSAP + 10
System Call Trap	NPSAP + 26	NPSAP + 28, 30	NPSAP + 12	NPSAP + 14
Segment Error Trap	NPSAP + 34	NPSAP + 36, 38	Unused	Unused
Nonmaskable Interrupt	NPSAP + 42	NPSAP + 44, 46	NPSAP + 20	NPSAP + 22
Nonvectored Interrupt	NPSAP + 50	NPSAP + 52, 54	NPSAP + 24	NPSAP + 26
Vectored Interrupt (V = Vector #)	NPSAP + 58	NPSAP + 4V + 60, 62	NPSAP + 28	NPSAP + 2V + 30



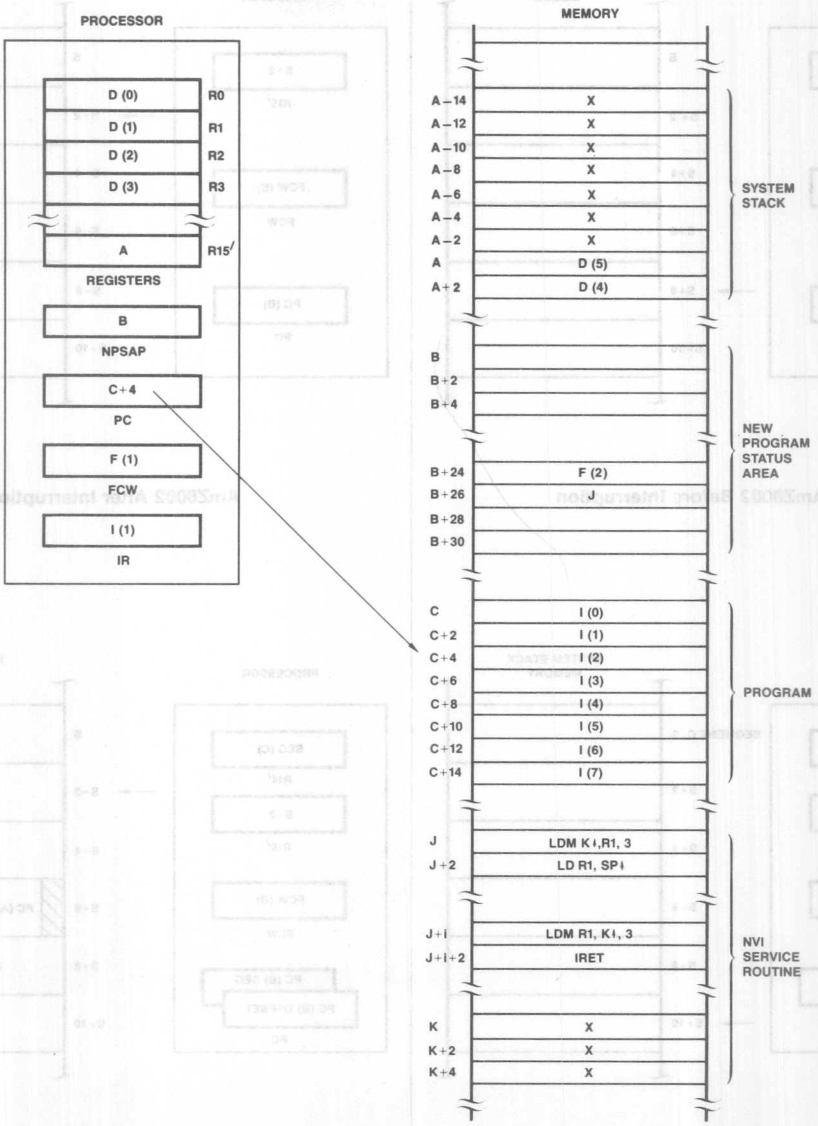
AmZ8002 Before Interruption

AmZ8002 After Interruption

AmZ8001 Before Interruption

AmZ8001 After Interruption

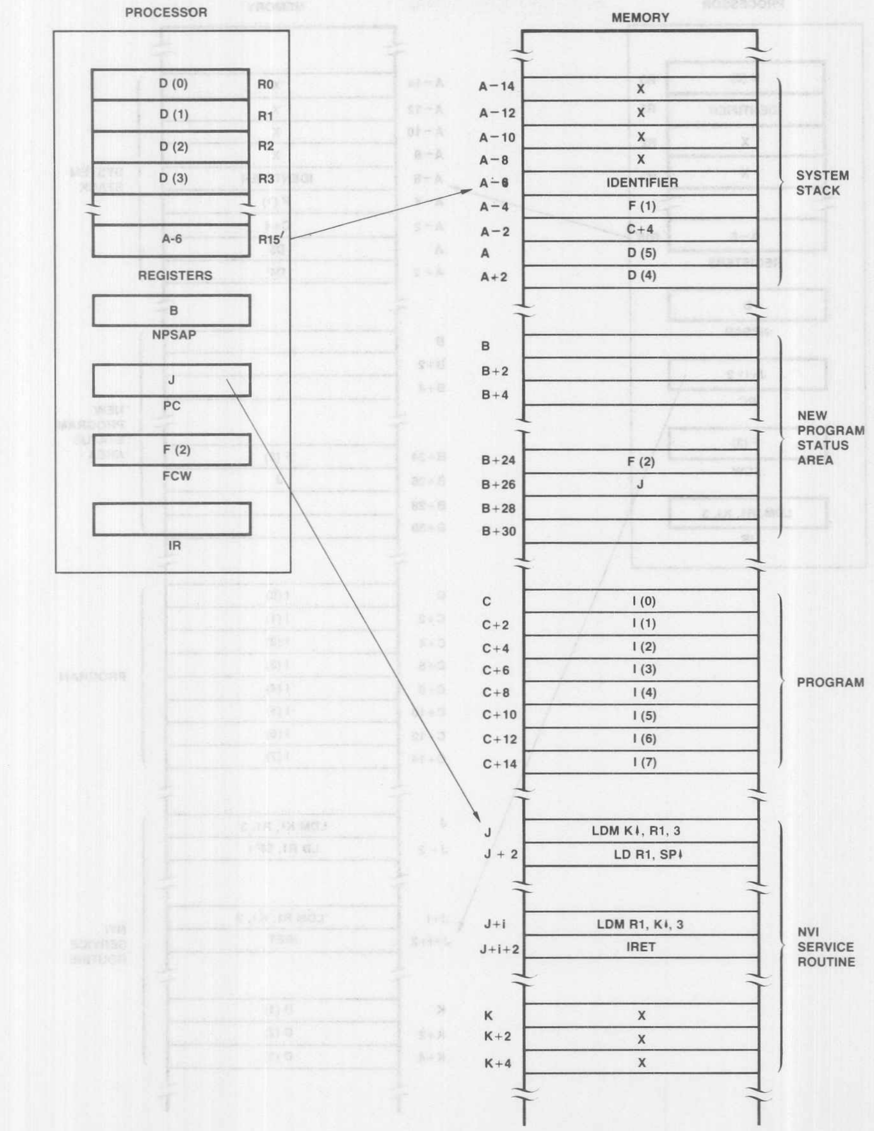
Figure 2.4.4.1 Program Status Changes



A. Status at the Time Interrupt is Received: Program Instruction I(1) is being executed. Interrupt example is NVI, and service routine will use registers R1, R2, and R3.

A, B, C	= Addresses in memory
I	= Instruction
D	= Data Item
J, K	= Addresses of service routine code and data
F	= Flag control word
X	= Don't care

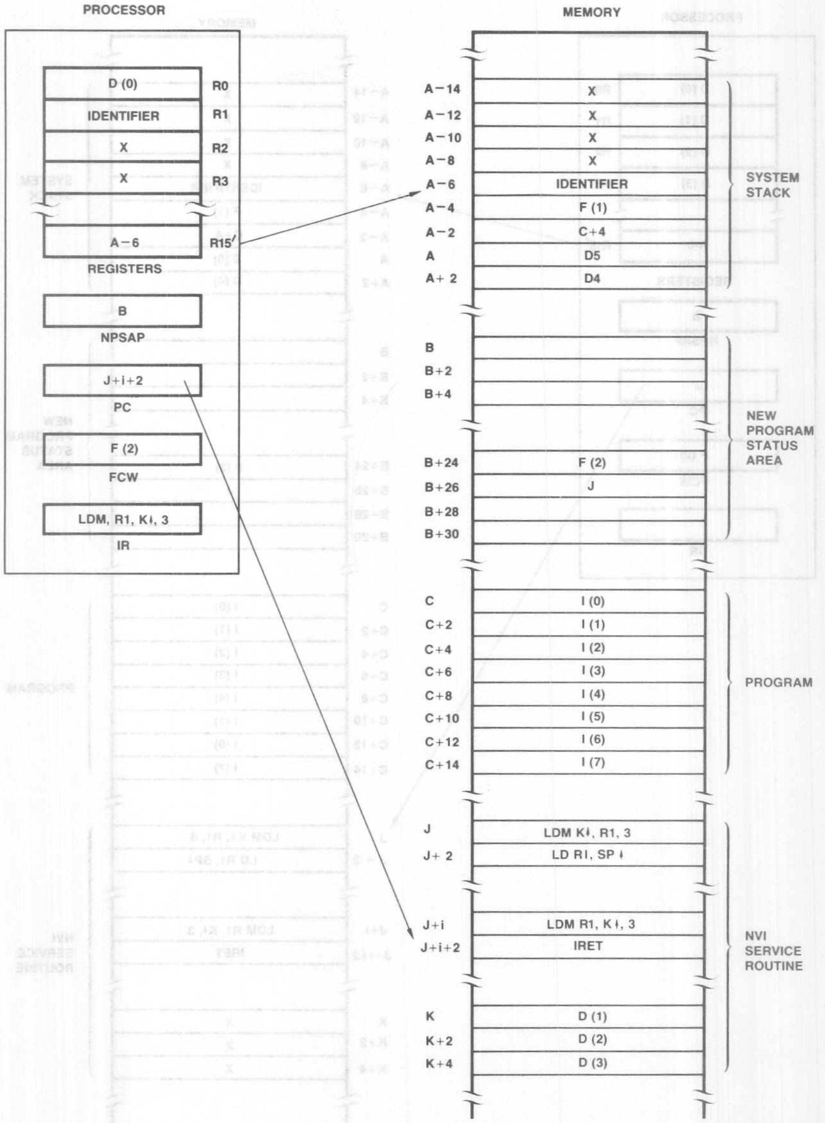
Figure 2.4.4.2 Exception Processing Sequence



B. Status after Interrupt is Acknowledged: Next instruction fetch will be at J. That instruction will save R1, R2, and R3 at K. The next instruction will copy the identifier into R1.

A, B, C	= Addresses in memory
I	= Instruction
D	= Data Item
J, K	= Addresses of service routine code and data
F	= Flag control word
X	= Don't care

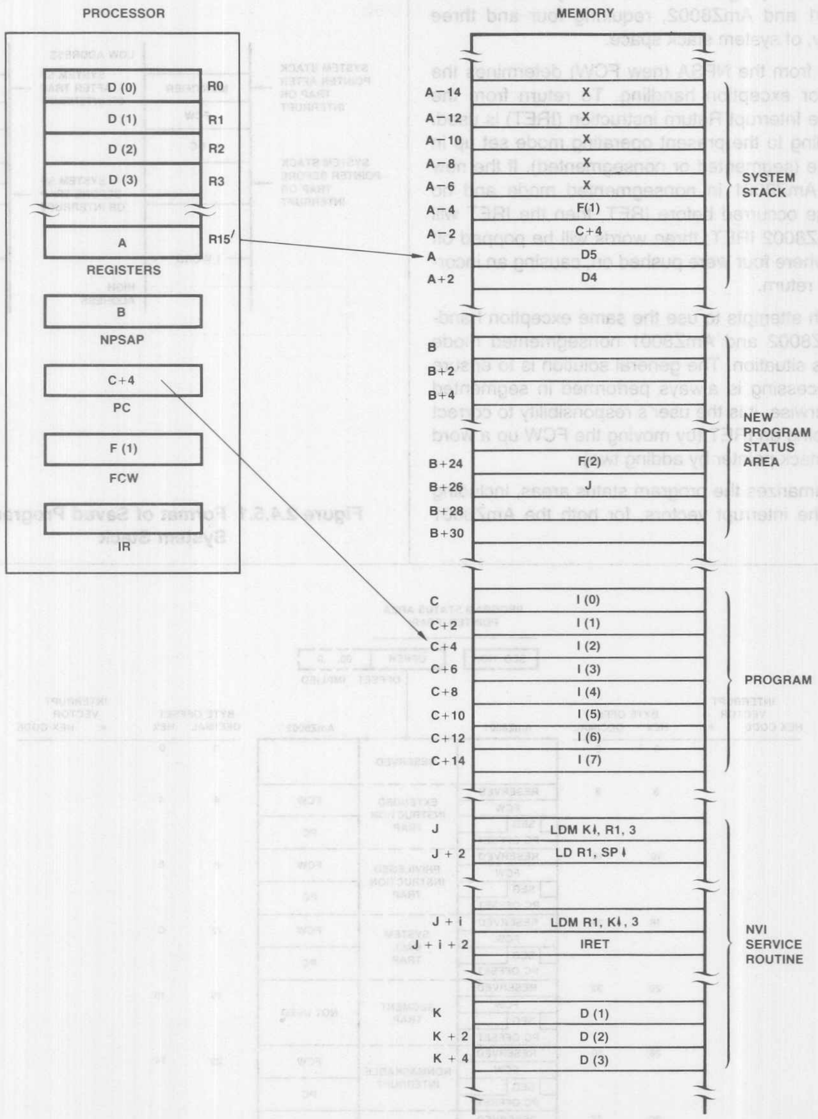
Figure 2.4.4.2 Exception Processing Sequence (Cont.)



C. Status Prior to End of Interrupt: Current Instruction will store registers; next instruction will return.

A, B, C	= Address in memory
I	= Instruction
D	= Data Item
J, K	= Addresses of service routine code and data
F	= Flag control word
X	= Don't care

Figure 2.4.4.2 Exception Processing Sequence (Cont.)



D. Status after Return from Interrupt: Next instruction fetch will be I(2) at C + 4.

A, B, C = Addresses in memory
I = Instruction
D = Data Item
J, K = Addresses of service routine code and data
F = Flag control word
X = Don't care

Figure 2.4.4.2 Exception Processing Sequence (Cont.)

2.4.5 AmZ8001 and AmZ8002 Exception Processing

The AmZ8001 always handles interrupt and trap processing by entering the segmented system mode, regardless of the FCW control bit settings for these two modes. Figure 2.4.5.1 shows the format of the saved program status in the system stack for both the AmZ8001 and AmZ8002, requiring four and three words, respectively, of system stack space.

The status loaded from the NPSA (new FCW) determines the operating mode for exception handling. To return from the interrupt routine the Interrupt Return instruction (IRET) is used. It functions according to the present operating mode set up in the interrupt routine (segmented or nonsegmented). If the new FCW placed the AmZ8001 in nonsegmented mode and no subsequent change occurred before IRET, then the IRET will function as an AmZ8002 IRET; three words will be popped off the system stack where four were pushed on, causing an incorrect context switch return.

Any software which attempts to use the same exception handling code for AmZ8002 and AmZ8001 nonsegmented mode may encounter this situation. The general solution is to ensure that exception processing is always performed in segmented mode. If done otherwise, it is the user's responsibility to correct the stack before doing an IRET (by moving the FCW up a word and adjusting the stack pointer by adding two).

Figure 2.4.5.2 summarizes the program status areas, including vector codes for the interrupt vectors, for both the AmZ8001 and AmZ8002.

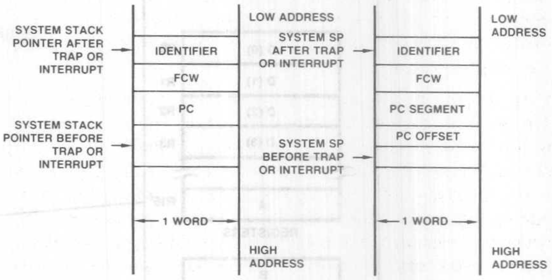

SYSTEM STACK POINTER AFTER TRAP OR INTERRUPT
SYSTEM STACK POINTER BEFORE TRAP OR INTERRUPT
IDENTIFIER
FCW
PC
LOW ADDRESS
SYSTEM SP AFTER TRAP OR INTERRUPT
SYSTEM SP BEFORE TRAP OR INTERRUPT
1 WORD
HIGH ADDRESS
IDENTIFIER
FCW
PC SEGMENT
PC OFFSET
1 WORD
LOW ADDRESS
HIGH ADDRESS

Figure 2.4.5.1 Format of Saved Program Status in the System Stack

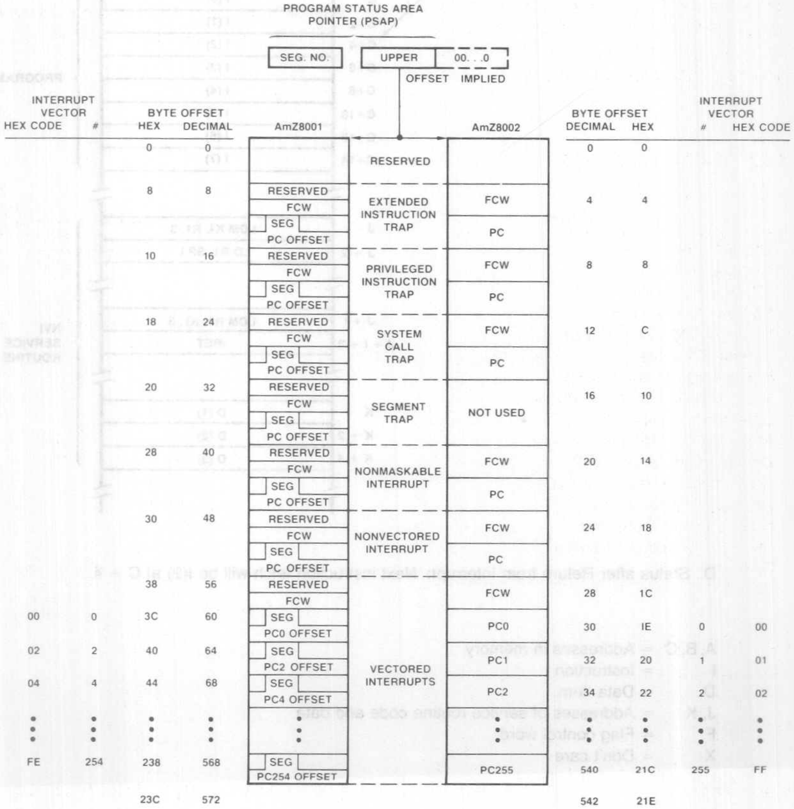


Figure 2.4.5.2 Program Status Area

2.5 STATUS LINES

The AmZ8000 CPU outputs status information over its four status lines (ST0-ST3) and the System/Normal line ($S/\overline{N}$). This information can be used to extend the addressing range or to protect accesses to certain portions of memory. The types of status information and their codes are listed in Table 2.5.

Status conditions are mutually exclusive and can, therefore, be encoded without penalty. Most status definitions are self-explanatory. One code is reserved for future enhancements of the AmZ8000 Family.

Extension of the addressing range is accomplished in an AmZ8000 system by allocating physical memory to specific usage (code vs. data space, for example) and using external circuitry to monitor the status lines and select the appropriate memory space for each address. For example, the direct addressing range of the AmZ8002 CPU is limited to 64K bytes; however, a system can be configured with 128K bytes if additional logic is used, say, to select the lower 64K bytes for program references and the upper 64K bytes for data references.

Protection of memory by access types is accomplished similarly. The memory is divided into blocks of locations and associated with each block is a set of legal status signals. For each access to the memory, the external circuit checks whether the CPU status is appropriate for the memory reference. The AmZ8010 Memory Management Unit is an example of an external memory-protection circuit, and it is discussed in section 2.2.4.

The first word in an instruction fetch has its own dedicated status code, namely 1101. This allows the synchronization of external circuits to the CPU. During all subsequent fetch cycles within the same instruction (the longest instruction requires a total of four word fetches), the status is changed from 1101 to 1100. Load Relative and Store Relative also have a status of 1100 with the data reference, so information can be moved from program space to data space.

TABLE 2.5 STATUS DECODES

ST3-ST0	Definition
0000	Internal operation
0001	Memory refresh
0010	I/O reference
0011	Special I/O reference
0100	Segment trap acknowledge
0101	Nonmaskable interrupt acknowledge
0110	Nonvectored interrupt acknowledge
0111	Vectored interrupt acknowledge
1000	Data memory request
1001	Stack memory request
1010	Data memory request (EPU)
1011	Stack memory request (EPU)
1100	Instruction space access
1101	Instruction fetch, first word
1110	Extension processor transfer
1111	Reserved

2.6 MEMORY AND I/O ADDRESSING

Like most 16-bit microprocessors, the AmZ8000 CPU uses a 16-bit parallel data bus between the CPU and memory or I/O. The CPU is capable of reading or writing a 16-bit word with every access. Words are always addressed with even addresses (A0 = 0). All instructions are words or multiple words and are aligned on even byte boundaries of memory.

The AmZ8000 CPU can, however, also read and write 8-bit bytes, so memory and I/O addresses are always expressed in bytes. The Byte/Word ($B/\overline{W}$) output indicates whether a byte or word is addressed (High = byte). A0 distinguishes between the upper and lower byte in memory or I/O. The most significant byte of the word is addressed when A0 is Low. Refer to Figure 2.6.

For word operations in both the read and write modes, $B/\overline{W}$ = Low, A0 is simply ignored and A1-A15 address the memory or I/O. For byte operations in the read mode, $B/\overline{W}$ = High, A0 is again ignored, and a whole word (both bytes) is read, but the CPU internally selects the appropriate byte. For byte operations in the write mode, the CPU outputs identical information on both the low (AD0-AD7) and the high (AD8-AD15) bytes of the Address/Data bus. External hardware must be used to enable writing in one memory byte and at the same time disable writing in the other byte, as defined by A0. The replication of byte information for writes is for current implementation and may change for subsequent AmZ8000 CPUs; therefore system designs should not depend upon this feature.

I/O transfers between CPU and I/O devices are peformed with 8- or 16-bit transfers. I/O devices are addressed with a 16-bit I/O port address. (Segment addresses are not involved.) The I/O port address is similar to a memory address; however, I/O address space is not part of the memory address space. I/O port and memory addresses co-exist on the same bus and they are distinguished by the status outputs.

Two types of I/O instructions are available: standard and special. Each has its own address space. Standard I/O instructions include a comprehensive set of In, Out and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status outputs distinguish between standard and special I/O references.

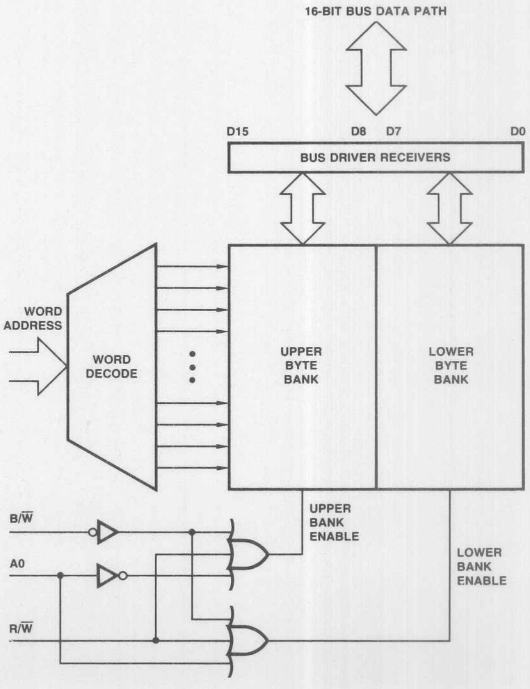


Figure 2.6 Byte/Word Selection of Memory

2.7 TIMING

Figure 2.7 shows the three basic timing periods of the AmZ8000: a clock cycle, a bus transaction, and a machine cycle. A *clock cycle* (sometimes called a T-state) is one cycle of the CPU clock, starting with a rising edge. A *bus transaction* covers a single data movement on the CPU bus and will last for three or more clock cycles, starting with a falling edge of $\overline{AS}$ and ending with a rising edge of $\overline{DS}$. A *machine cycle* covers one basic CPU operation and always starts with a bus transaction. A machine cycle can extend beyond the end of a transaction by an unlimited number of clock cycles.

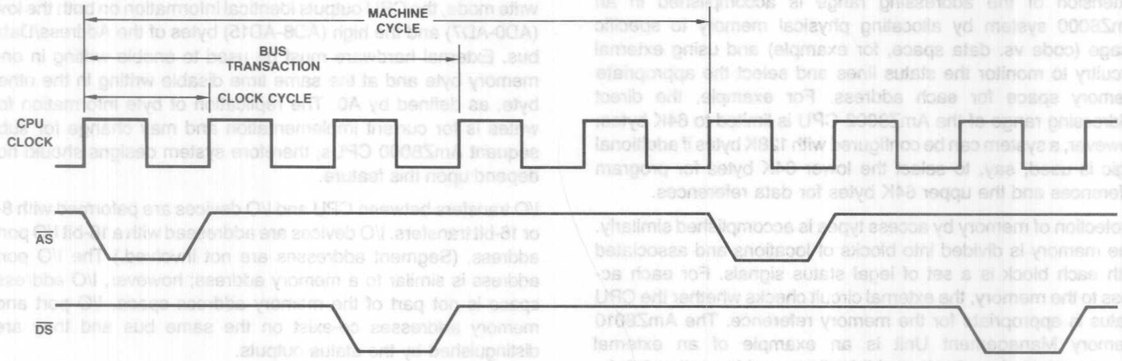


Figure 2.7 Basic Timing Periods

3.0 ADDRESSING AND DATA ORGANIZATION

3.1 INTRODUCTION

Operands in the AmZ8000 may be part of the instruction, in registers, or in memory. In any case they may be bytes, words or long words. For operands in memory they may also be addressed as strings – sequences of bytes or words up to 64K bytes long. The type of operand, byte, word, long-word or string, is determined by the instruction. Most instructions have both byte and word forms. (The mnemonic for the word form of the instruction is the basic name. The byte form appends a B to the name and the long-word form appends an L.) For each instruction there is one or more possible address modes used to designate the location of the operand. The general form of the various modes and detailed descriptions follow later. Addressing modes are also discussed in detail.

3.2 ADDRESSING DATA

Data can be addressed and stored in the general-purpose registers, in memory, or in instructions. Because the registers are general-purpose, addresses can also be easily manipulated as data. Bottlenecks due to information exchanges between dedicated data and address registers do not exist.

3.2.1 Addressing Data in Registers

Instructions refer to data in registers using the **R** (Register) address mode. The opcode specifies byte, word or long word, and the appropriate register or register pair is referenced as shown in Table 2.3.1 and Table 3.2.1.

3.2.2 Addressing Data in Memory

Data located in memory is referenced by supplying in the instruction one of the following:

a. The complete address – **DA** (direct address) mode.

b. The name of a register or register pair containing the complete address – **IR** (indirect register) mode.

c. A complete address and the name of a register whose contents is to be added to the specified address – **X** (indexed) mode.

d. A displacement and the name of a register or register pair containing a complete address to which the displacement should be added to that address – **BA** (base indexed) mode.

e. The name of a register or pair containing a complete address and the name of a register containing a displacement to be added to that address – **BX** (base indexed) mode.

f. A displacement which is to be added to the Program Counter – **RA** (relative address) mode.

g. Two registers, one containing the address of the beginning or end of a series of data items and the other containing the length of the series. Following each iteration of this type of instruction the address is changed to point to the next item and the count is decremented. This type of addressing is used for string manipulation and block I/O. Some instructions use a third register which points to a second string of data items to be processed with the first.

TABLE 3.2.1 DATA ADDRESSING IN REGISTERS

	Register	Byte Operand		Word Operand		Long-Word Operand	
HEX	Designation	Name	Data In	Name	Data In	Name	Data In
0	0000	RH0	R0<8:15>	R0	R0	RR0	D<16:31> in R0 D<0:15> in R1
1	0001	RH1	R1<8:15>	R1	R1		Reserved
2	0010	RH2	R2<8:15>	R2	R2	RR2	D<16:31> in R2 D<0:15> in R3
3	0011	RH3	R3<8:15>	R3	R3		Reserved
4	0100	RH4	R4<8:15>	R4	R4	RR4	D<16:31> in R4 D<0:15> in R5
5	0101	RH5	R5<8:15>	R5	R5		Reserved
6	0110	RH6	R6<8:15>	R6	R6	RR6	D<16:31> in R6 D<0:15> in R7
7	0111	RH7	R7<8:15>	R7	R7		Reserved
8	1000	RL0	R0<0:7>	R8	R8	RR8	D<16:31> in R8 D<0:15> in R9
9	1001	RL1	R1<0:7>	R9	R9		Reserved
A	1010	RL2	R2<0:7>	R10	R10	RR10	D<16:31> in R10 D<0:15> in R11
B	1011	RL3	R3<0:7>	R11	R11		Reserved
C	1100	RL4	R4<0:7>	R12	R12	RR12	D<16:31> in R12 D<0:15> in R13
D	1101	RL5	R5<0:7>	R13	R13		Reserved
E	1110	RL6	R6<0:7>	R14	R14	RR14	D<16:31> in R14 D<0:15> in R15
F	1111	RL7	R7<0:7>	R15	R15 in normal mode R15′ in system mode (SP for AmZ8002)	RR14′	Data in R14′ and R15′ in system mode (SP for AmZ8001)

(Register quads are not shown in the table. They are designated as RQ0, RQ4, RQ8, and RQ12.)

3.2.3 Data Storage in Memory

Memory address space is viewed to a chain of consecutively numbered (in ascending order) bytes, as shown in Figure 3.2.3.1. The number of each byte is its address, and the byte is the basic addressable element. A word spans two bytes, and is addressed by the address of its high order byte (most significant), with the lowest absolute address of the two bytes, which is always an even address. A long-word consists of four bytes and is also referred by the address of its high order byte (most significant word), which is the lowest absolute address of the four bytes.

Instructions and addresses stored in memory are always on word boundaries; they always have even numbered addresses.

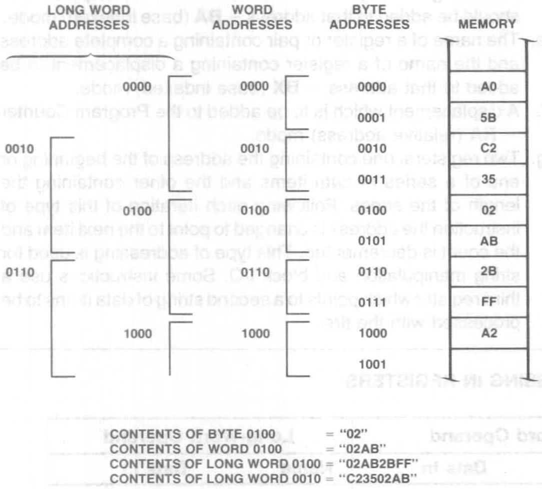


Figure 3.2.3.1 Data Storage in Memory

Note that this format differs from the PDP-11 but is identical to the IBM convention. The reason for choosing this format is because the AmZ8000 CPU can operate on 32-bit long words and also on byte and word strings. It is important to maintain a continuity of order when words are concatenated into long words and strings. Making ascending addresses proceed from the highest byte of the first word to the lowest byte of the last word maintains this continuity, and allows comparing and sorting of byte and word strings. Refer to Figure 3.2.3.2.

String instructions, such as I/O and block compare, refer to a series of bytes or words in consecutive memory locations. Both autoincrement and autodecrement forms of these instructions exist, so the string can be scanned starting at either end. The byte form of these instructions modifies the address by one on each iteration to point to the next byte; the word form modifies the address by two to point to the next word.

Bit labeling within a byte does not follow this order. The least significant bit in a byte, word or long word is called Bit 0 and occurs in the byte with the highest memory address. This is consistent with the convention where bit n corresponds to position 2^n in the conventional binary notation. This ordering of bit numbers is also followed in the registers.

3.2.4 Data Contained in Instruction

Most instructions allow an operand to be contained within the instruction itself. This is the **IM** (immediate) address mode. Generally, the data follows the opcode and any addresses which are also part of the instruction.

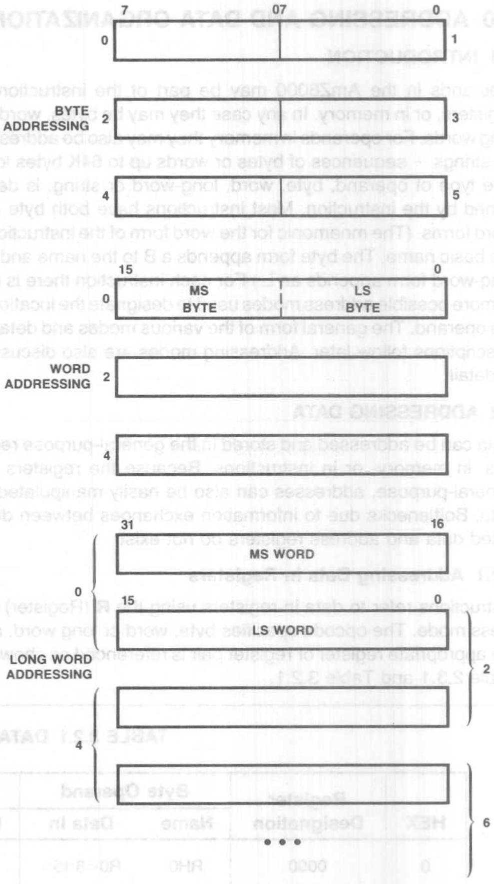


Figure 3.2.3.2 Memory Addressing

Immediate data may consist of a 16-bit word, long-word, or a byte. The long-word has the more significant 16 bits first, then the less significant 16 bits in the second word. A byte operand uses a full 16-bit word with the same data in both the upper and lower bytes.

There are a few instructions which permit immediate data to be located directly in the single 16-bit word which contains the opcode, forming a one-word instruction. These include a Load Byte instruction, and Increment or Decrement by any integer from 1 through 16.

3.3 MEMORY ADDRESS FORMATS

A complete memory address may take one of three forms. Refer to Figure 3.3.

3.3.1 NS (Nonsegmented)

A single 16-bit word which specifies the address in the AmZ8002 or in the nonsegmented mode of the AmZ8001.

3.3.2 SSO (Segmented Short Offset)

A single 16-bit word in the form shown having a shortened offset address, but including a segment number.

This form can address a byte in the first 256 bytes of any segment. The range is effectively extended to the entire memory space when this form is used with the indexed addressing mode. This form offers the ability to access data in any segment of memory without using two full words to supply the address.

The short offset segmented address can be used only in direct address and indexed addressing modes where the address is part of the instruction.

3.3.3 SLO (Segmented Long Offset)

Two 16-bit words in the form shown completely specifying both the 16-bit offset and the 7-bit segment addresses.

This form can refer directly to any byte in the memory. The word containing the segment always precedes the word containing the offset. When a register pair is used to hold this form, the segment is in the even numbered register and the offset is in the the odd numbered register of the pair. (For example, in RR2, R2 contains the segment and R3 contains the offset.)

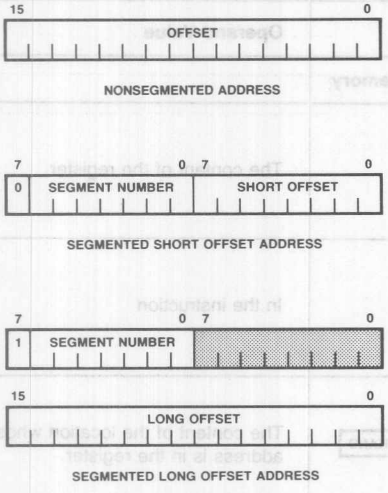


Figure 3.3.1 Memory Addressing Formats

In the general unrestricted case of long offset, the segmented address occupies two words, as described before. The most significant bit in the segment word is a one in this case.

The short offset mode squeezes the segment number and offset into one word, saving program size and execution time. Since 23 bits obviously do not fit into a 16-bit word, the eight most significant bits of the offset are omitted and implied to be zero. The most significant bit of the address word is made zero to indicate short offset mode. Short offset addresses are thus limited to the first 256 bytes at the beginning of each segment. At first this may appear to be a restriction, but it is very useful when used with the index mode, where the index register can always supply the full 16-bit range of the offset. Short offset saves one instruction word and speeds up execution by two clock cycles in direct address mode and three clock cycles in indexed mode.

3.4 SEGMENTED ADDRESS FORMATS

Figure 3.4 shows the format for segmented addresses.

3.5 DATA TYPES

Operands are 1, 4, 8, 16, 32 or 64 bits, as specified by the instruction. In addition, strings of 8- or 16-bit data can be manipulated by single instructions. Of particular interest are the increased precisions of the arithmetic instructions. Add and Subtract instructions can operate on 8-, 16-, or 32-bit operands; Multiply instructions can operate on 16-bit or 32-bit multiplicands; and Divide instructions can operate on 32- or 64-bit dividends. The Shift instructions can operate on 8-, 16-, and 32-bit registers.

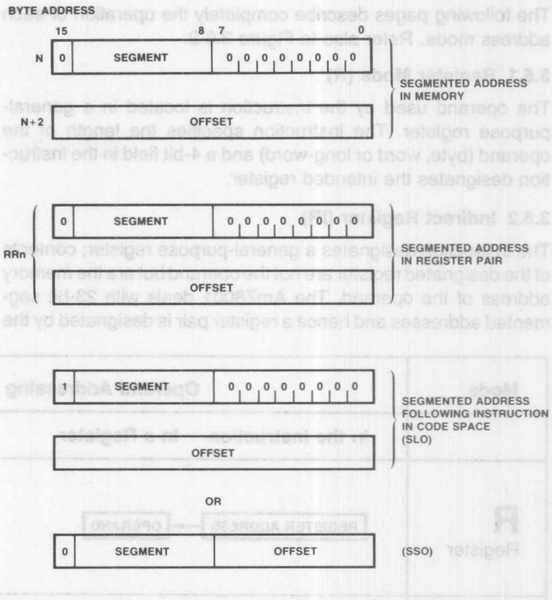


Figure 3.4 Segmented Address Formats

3.6 ADDRESS MODES

In addition to supporting the primitive operands of bits, digits, bytes, 16- and 32-bit integers, and byte and word strings, the AmZ8000 CPU's rich set of addressing modes supports high-level data constructs such as arrays, lists, and records. These combine with powerful instructions to significantly extend the capabilities of microprocessors.

The address mode for a given instruction is determined by certain bits in the instruction, as shown in Figure 3.6.1. Refer to the format and decoding sections of the instruction set organization chapter.

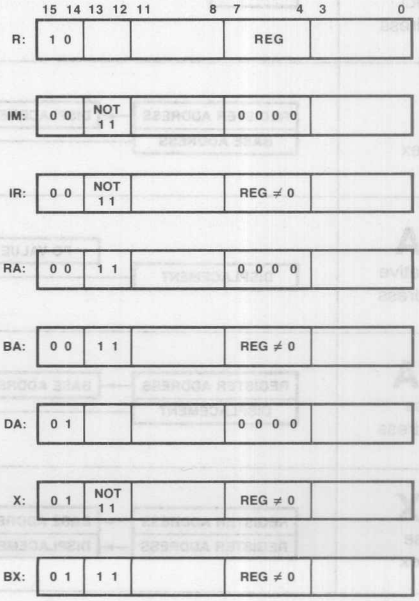


Figure 3.6.1 Decoding Address Modes

The following pages describe completely the operation of each address mode. Refer also to Figure 3.6.2.

3.6.1 Register Mode (R)

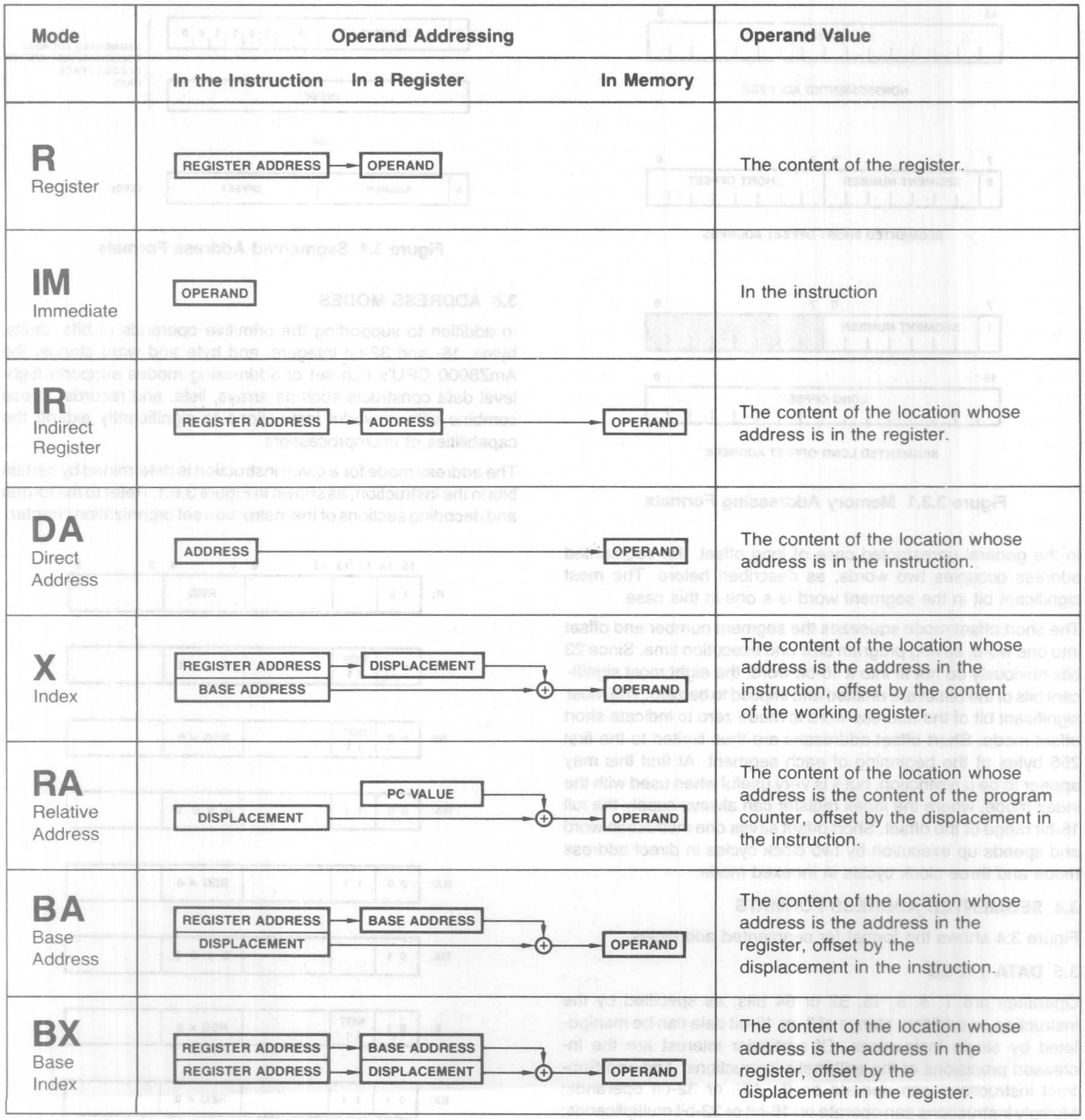
The operand used by the instruction is located in a general-purpose register. The instruction specifies the length of the operand (byte, word or long-word) and a 4-bit field in the instruction designates the intended register.

3.6.2 Indirect Register (IR)

The instruction designates a general-purpose register; contents of the designated register are not the operand but are the memory address of the operand. The AmZ8001 deals with 23-bit segmented addresses and hence a register pair is designated by the instruction (in segmented mode). The first register contains the 7-bit segment number and the second register contains the 16-bit offset. Any general-purpose register pair except RR0 can be designated for this addressing mode. The AmZ8002 requires only 16-bit addresses and hence any general-purpose register except R0 can be designated for IR addressing mode. (Some exceptions to this are noted in the instruction set description. See also Autoincrement and Autodecrement.)

3.6.3 Direct Address (DA)

The instruction itself explicitly specifies an address and the operand used by the instruction is located at that address. In the AmZ8001, direct addresses are specified in one of two formats long offset and short offset. For the long offset, the memory word



Mode	Operand Addressing			Operand Value
	In the Instruction	In a Register	In Memory	
R Register	REGISTER ADDRESS	OPERAND		The content of the register.
IM Immediate	OPERAND			In the instruction
IR Indirect Register	REGISTER ADDRESS	ADDRESS	OPERAND	The content of the location whose address is in the register.
DA Direct Address	ADDRESS		OPERAND	The content of the location whose address is in the instruction.
X Index	REGISTER ADDRESS BASE ADDRESS	DISPLACEMENT	OPERAND	The content of the location whose address is the address in the instruction, offset by the content of the working register.
RA Relative Address	DISPLACEMENT	PC VALUE	OPERAND	The content of the location whose address is the content of the program counter, offset by the displacement in the instruction.
BA Base Address	REGISTER ADDRESS DISPLACEMENT	BASE ADDRESS	OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the instruction.
BX Base Index	REGISTER ADDRESS REGISTER ADDRESS	BASE ADDRESS DISPLACEMENT	OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the register.

Figure 3.6.2 Addressing Modes

immediately following the instruction opcode word contains the 7-bit segment number, and the memory word immediately following the segment number word is the 16-bit offset. For the short offset, the memory word immediately following the instruction opcode word contains both 7-bit segment number and 8-bit offset. In the AmZ8002 the memory word immediately following the instruction opcode word contains the 16-bit address.

3.6.4 Immediate Mode (IM)

The instruction itself contains the operand. In certain short instructions the operand and opcode are in one word. In general, the operand is in the last word or words of the instruction. Byte operands are repeated in both halves of the word.

3.6.5 Indexed Mode (X)

The instruction designates a 16-bit general-purpose register as the index register. Any general-purpose register except R0 can be used as the index register. The instruction also specifies an address as in the direct address mode. In the AmZ8001 the 16-bit contents of the designated index register are added to the 16-bit offset value specified in the instruction. Both index and offset are treated as 16-bit unsigned integers and any carry from the most significant bit position during this addition is ignored. The resulting 16-bit sum together with the 7-bit segment number specified in the instruction is used as 23-bit segmented address. The operand will be located at this address in memory. If short addressing offset is used in the AmZ8001 for indexed addressing mode, the memory word immediately following the instruction opcode word contains both a 7-bit segment number and an 8-bit offset.

A 16-bit unsigned integer is formed whose least significant byte is the 8-bit offset specified and most significant byte is zero. The 16-bit word thus formed is added to the 16-bit unsigned integer contained in the designated general-purpose register. Any carry from the most significant bit position during this addition is ignored. The 16 bits resulting from this addition together with the 7-bit segment number specified is the 23-bit address. The operand will be located in the memory at this address.

In the AmZ8002 the memory word immediately following the instruction opcode word contains a 16-bit address. This unsigned integer is added to the 16-bit unsigned integer located in the designated index register. The carry from the most significant bit position during this addition is ignored. The resulting 16-bit address is where the operand is located in the memory.

3.6.6 Base Address Mode (BA)

The instruction designates a general-purpose register as the base address register. In the case of the AmZ8001 the instruction designates a register pair such that the 7-bit segment number is contained in one register and the 16-bit offset is contained in the other as shown. In the case of the AmZ8002 the designated base address register contains a 16-bit address. Any general-purpose register except R0 or register pair except RR0 can be designated as the base address register. The memory word immediately following the instruction opcode word contains a 16-bit displacement. Both displacement and base address are treated as unsigned binary integers. The 16-bit displacement is added to the 16-bit base address (16-bit offset in the AmZ8001) and carry occuring from the most significant bit position during this addition is ignored. The resulting 16-bit value (together with the segment number of the base address in the AmZ8001) is the address of the operand in memory.

3.6.7 Base Indexed Mode (BX)

The instruction designates a general-purpose register (register pair in AmZ8001) as the base address register. The instruction also designates a 16-bit general-purpose register as displacement. Any general-purpose register except R0 (AmZ8002) or any register pair except RR0 (AmZ8001) can be used as the base address register. Similarly any general-purpose register except R0 can be used as the displacement register. Both base address and displacement are unsigned integers.

The 16-bit displacement is added to the base address (or offset of the base address in AmZ8001) and carry from the most significant bit position during this addition is ignored. The 16-bit result (together with base address segment number) is the address of the operand in memory.

3.6.8 Relative Address (RA)

The instruction itself contains a displacement. This displacement is a signed integer using two's complement notation. The number of bits allocated to represent the displacement depend on the instruction where relative addressing mode is available. The displacement is sign extended appropriately to obtain a signed 16-bit displacement. The sign extended displacement is added to the 16-bit program counter (PC offset in AmZ8001). Carry from the most significant bit position during this addition is ignored. As soon as the instruction using the relative address mode is fetched, the PC will be updated. Hence, the updated PC value (i.e., address of the following instruction) will be used for address calculations.

The 16-bit value obtained by adding the PC and displacement (together with the segment number in AmZ8001) is the address of the operand in memory.

3.6.9 Autoincrement and Autodecrement

These two implied addressing modes are only used in string manipulation instructions. These addressing modes are a variation of the IR addressing mode. The instruction designates a general-purpose register (or a register pair in AmZ8001) whose contents are used as the address. After fetching the operand the contents of the register are incremented or decremented depending on Autoincrement or Autodecrement. In the case of the AmZ8001, only the register containing the offset is affected and any carry resulting from this operation is ignored. For byte operations, incrementing or decrementing by one occurs. For word operations, incrementing or decrementing by two takes place. R0 and RR0 can be designated in the IR addressing mode for the autoindexing instruction.

3.6.10 Port Addressing Modes

Input/output transfers between CPU and I/O devices are performed with 8- or 16-bit transfers. The address of the I/O ports is similar to a memory address in that they coexist on the same bus; however, they are in separate address spaces distinguished by the status outputs.

I/O devices are addresses with a 16-bit I/O port address. Segment addresses are not involved. Two types of I/O instructions are supported with separate I/O spaces distinguished by the status outputs. These instructions use two types of port addressing modes:

Port register **(PR)**: The instruction designates a general-purpose register (never a register pair) which contains the port address (similar to IR memory addressing mode).

Port Address **(PA)**: The instruction contains an explicit port address word which specifies the I/O port address (similar to DA memory addressing mode).

The autoindexing versions of the I/O instructions combine port register port addressing with the indirect register memory addressing mode.

4.0 INSTRUCTION SET ORGANIZATION

4.1 INTRODUCTION

The AmZ8000 offers an abundant instruction set that represents a major advance over its predecessors. The Load and Exchange instructions have been expanded to support operating system functions and conversion of existing microprocessor programs. The usual Arithmetic instructions can now deal with higher-precision operands, and hardware Multiply and Divide instructions have been added. The Bit Manipulation instructions can access a calculated bit position within a byte or word, as well as specify the position statically in the instruction.

The Rotate and Shift instructions are considerably more flexible than those in previous microprocessors. The String instructions are useful in translating between different character codes. Special I/O instructions are included to manage peripheral devices, such as the Memory Management Unit (AmZ8010), that do not respond to regular I/O commands. Multiple-processor configurations are supported by special instructions.

The following instructions exemplify the innovative nature of the AmZ8000 instruction set. A complete list of AmZ8000 instructions can be found in the Instruction Set section.

Load and Exchange Instructions

Exchange Byte (EX) is practical for converting Z-80, 8080, 6800 and other microprocessor programs into AmZ8000 code, because the AmZ8000 uses the opposite assignment of odd/even addresses in 16-bit words.

Load Multiple (LDM) saves n registers and is useful for switching tasks.

Load Relative (LDR) loads fixed values from program space into data space.

Arithmetic Instructions

Add With Carry and Subtract With Carry (ADC, SBC) are conventionally used in 8-bit microprocessors for multiprecision arithmetic operations. These instructions are rarely used with the AmZ8000 CPU because it has 16- and 32-bit arithmetic instructions.

Decrement By N and Increment By N (DEC, INC) are intended for address and pointer manipulation, but can also be used for Quick Add/Subtract Immediate with 4-bit nibbles. The flag setting is different from Add/Subtract instructions – as is conventional – in that the Carry and Decimal Adjust flags are unaffected by the Increment and Decrement instructions to support multiple precision arithmetic.

Decimal Adjust (DAB) automatically generates the proper 2-digit BCD result after a byte Add or Subtract operation, and eliminates the need for special decimal arithmetic instructions.

Multiply (MULT) provides signed (two's complement) multiplication of two words, generating a long-word result; or of two long-words generating a quadruple word result. No byte multiply exists because it is rarely used and, after sign extension, can be performed by a word multiply.

Divide (DIV) provides signed (two's complement) division of a long word by another word, generating a word quotient and a remainder word; or of one quadruple-word by a long-word, generating a long-word quotient and long-word remainder.

Both Multiply and Divide use a conforming register assignment. That is, a multiply followed by a divide on the same registers is essentially a no-op. The register designation used in the operating description must be even for word operations and must be a multiple of four for long-word operations.

Logical Instructions

Test Condition Code (TCC) performs the same test as a Jump instruction, but affects the least significant bit of a specified register instead of changing the PC.

Program Control Instructions

Call Relative (CALR) is a shorter, faster version of Call, but with a limited range.

Decrement And Jump If Non-Zero (DJNZ) is a one-word basic looping instruction.

Jump Relative (JR) is a shorter, faster version of Jump, but with limited range.

Bit Manipulation Instructions

Test Bit, Reset Bit, Set Bit (BIT, RES, SET) are available in two forms: static and dynamic. For the static form, any bit (the position is defined in the immediate word of the instruction) located in any byte or word in any register or in memory can be set, reset or tested (inverted and routed into the Z flag).

For the dynamic form, any bit (the position is defined by the content of a register that is, in turn, specified in the instruction) located in any byte or word in any register, but not in memory, can be set, reset or tested.

Test And Set (TSET) is a read/modify/write instruction normally used to create operating system locks. The most significant bit of a byte or word in a register or in memory is routed into the S flag bit and the whole byte or word is then set to all 1s. During this instruction, the processor does not relinquish the bus.

Test Multi-Micro Bit and Multi-Micro Request/Set/Reset (MBIT, MREQ, MSET, MRES) are used to synchronize the access by multiple microprocessors to a shared resource, such as common memory, bus, or I/O device.

Note that the instruction MREQ (Multi-Microprocessor Request) has nothing whatsoever in common with the $\overline{\text{MREQ}}$ (Memory Request) output from the AmZ8000 CPU.

Rotate and Shift Instructions

The AmZ8000 CPU has a complete set of shift instructions that shift any combination of bytes or words, right or left, arithmetically or logically, by any meaningful number of positions as specified either in the instruction (static) or in a register (dynamic).

The CPU also has a smaller repertoire of rotate instructions that rotates bytes or words, either right or left, through carry or not, and by one bit or by two bits.

The instructions Rotate Digit Left and Rotate Digit Right (RDLB, RRDB) rotate 4-bit BCD digits right or left, and are used in BCD arithmetic operations.

Block Transfer and String Manipulation Instructions

Translate And Decrement/Increment (TRDB, TRIB) is used for code conversion, such as ASCII to EBCDIC. These instructions translate a byte string in memory by substituting one string by its table-lookup equivalent. TRDB and TRIB execute one operation and decrement the contents of the length register; thus they are useful as part of loop performing several actions on each character.

Translate, decrement/Increment and Repeat (TRDBB, TRIRB) are the same as TRDB and TRIB, except they repeat automatically until the contents of the length register become zero. They are therefore useful in straightforward translation applications.

Translate And Test, Decrement/Increment (TRTDB, TRTIB) test a character according to the contents of the translation table.

Translate And Test, Decrement/Increment And Repeat (TRTDRB, TRTIRB) scans a string of characters. The first character is tested and, depending on the contents of the translation table, the process stops or skips to the next character. Stopped characters can be used for further processing.

I/O and Special I/O Instructions

The AmZ8000 CPU has two complete sets of I/O instructions: Standard I/O and Special I/O. The only difference is the status information on the ST0-ST3 outputs.

Standard I/O instructions are used to communicate with AmZ8000 bus compatible peripherals. Special I/O instructions are typically used for communicating with the Memory Management Unit.

Both types of instructions transfer eight or 16 bits and use a type of 16-bit addressing analogous to the AmZ8002 memory-addressing scheme: For word operations, A0 is always zero; in byte-input operations, A0 is used internally by the CPU to select the appropriate byte; in byte-output operations, the byte is duplicated in the high and low bytes of the address/data bus, and external logic uses A0 to enable the appropriate output device.

4.2 INSTRUCTION FORMAT

The CPU instructions are one to five words long, depending on the type of instruction and addressing mode. Instructions are located in memory and must be word aligned. The first word of an instruction always contains the opcode. Depending on the addressing mode, one or more words will follow the opcode word of an instruction. Figure 4.2 illustrates the general instruction word format. Some instructions contain fields that differ from the generalized format shown. All such variations can be ascertained by referring to the individual instruction descriptions found in later sections of this document.

4.3 INSTRUCTION DECODING

The Mode Field (bit 14 and bit 15), together with bits 12 and 13 and bits 4, 5, 6 and 7, determines the applicable addressing mode. Bit 8 of the opcode word specifies word or byte operand whenever applicable. Bits 4, 5, 6 and 7 normally designate a general-purpose register. Note that when designating a register pair, bit 4 must be zero and only 5, 6, and 7 are used. Refer to Figure 4.3. For Register mode of addressing there are no restrictions on the values of bits 4, 5, 6 and 7. Only the mode field is needed to specify this addressing mode for any general-purpose register. However, for IM, RA and DA addressing modes, bits 4, 5, 6 and 7 must all be zero. For these addressing modes, zeroes in bits 4, 5, 6 and 7 are not interpreted as general-purpose register number zero. Similarly, for IR, BA, X and BX addressing modes, bits 4, 5, 6 and 7 cannot be zero. In other words, general-purpose register number zero cannot be used in these addressing modes. It should be emphasized that if a register pair is needed for these addressing modes, bit 4 is always zero and the non-zero requirement applies to bits 5, 6 and 7. (See also Section 5.2.4.)

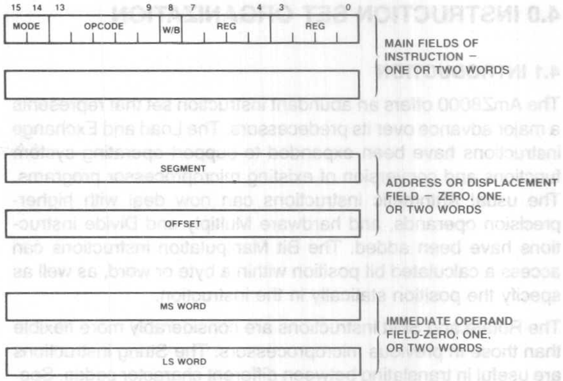

15
14
13
9
8
7
4
3
0
MODE
OPCODE
W/B
REG
REG
MAIN FIELDS OF INSTRUCTION – ONE OR TWO WORDS
SEGMENT
OFFSET
ADDRESS OR DISPLACEMENT FIELD – ZERO, ONE, OR TWO WORDS
MS WORD
LS WORD
IMMEDIATE OPERAND FIELD-ZERO, ONE, OR TWO WORDS

Figure 4.2 General Instruction Word Format

4.4 SEGMENTED AND NONSEGMENTED MODES

In the AmZ8002 addresses are completely contained in a single 16-bit word. The AmZ8001 uses a segment and an offset, requiring two words to contain the 23-bit segmented address.

All the instructions have both segmented and nonsegmented forms. The only difference is the address references. The AmZ8001 has both segmented and nonsegmented modes of

Address Mode

Bits 15, 14	Bits 13, 12	Bits 7, 6, 5, 4	Mode
00	Not 11	0	IM
00	Not 11	Not 0	IR
00	11	0	RA
00	11	Not 0	BA
01	X	0	DA
01	Not 11	Not 0	X
01	11	Not 0	BX
10	X	X	R
11	X	X	Special*

*Used for short one-word instructions.

Word/Byte

Bit 8	Mode
0	Byte
1	Word

Address, When Present

Displacement	16-Bit Word
Nonsegmented Address	16-Bit Word
Segmented Short Offset	0 + 7-Bit Segment + 8-Bit Offset
Segmented Long Offset	First Word: 1 + 7-Bit Segment + 8-Bit Unused Second Word: 16-Bit Offset

Immediate Data, When Present

Byte	Same Byte in Both Halves of Word
Word	16-Bit Word
Long-Word	First Word: Bits 16:31 of Operand Second Word: Bits 0:15 of Operand

Figure 4.3 Instruction Decoding

operation. The segment bit (15) of the FCW is used to enable the segmented mode. In the nonsegmented mode the AmZ8001 emulates the AmZ8002 so that code assembled for the nonsegmented AmZ8002 will execute on the segmented AmZ8001.

Since the hardware will be designed for the segmented processor, the AmZ8001 will continue to supply segment addresses even though the instructions do not contain them. The segment number address will be whatever it was prior to the switch to the nonsegmented mode. All memory accesses will be in the PC segment space.

Refer to section 2.2.3 for a general discussion of segmented memory addressing. Refer to Table 4.4 to see the distinctions of segmented and nonsegmented modes with respect to the addressing modes and the number of instruction words required.

4.5 CONDITION CODES

The Condition Code (CC) is a 4-bit field in some instructions that specifies certain flag settings. The operation performed by the instruction is in most cases determined by the outcome of comparing the actual flag settings with that specified by the CC field. Instructions that specify CC field include conditional jumps, return from subroutine and block/string manipulating instructions. The Condition Code definitions consist of true and false settings of the C, Z and P/V flags, signed and unsigned comparisons as shown in Table 4.5. One of the CC values specifies unconditional combinations in which flag settings are ignored.

4.6 INPUT/OUTPUT INSTRUCTIONS

A set of input/output (I/O) instructions is provided to perform 16-bit or 8-bit transfers including block transfers between the CPU and I/O devices. Input/Output devices are addresses using a 16-bit

Table 4.4 SEGMENTED AND NONSEGMENTED MODES

		Nonsegmented Mode		Segmented Mode
		AmZ8002	AmZ8001	AmZ8001
PC, SP and NPSAP		Standard	Segmented	Segmented
IR	Address Register	Rn	Rn	RRn
DA	Address Field	Word	Word	Word (SSO), Long Word (SLO)
X	Address Field Displacement (Index) Register	Word Rn	Word Rn	Word (SSO), Long Word (SLO) Rn
BA	Displacement Field Base Address Register	Word Rn	Word Rn	Word RRn
BX	Base Address Register Displacement (Index) Register	Rn Rn	Rn Rn	RRn Rn
RA	Dispacement*	Word	Word	Word

*JR, CALR, DBJNZ, displacement field <16 bits.

TABLE 4.5 CC – FIELD DECODING

CC Field	Assembler Notation	Meaning	Flag Settings for CC True
1110	NZ	Not Zero	Z = 0
0110	ZR	Zero	Z = 1
1111	NC	No Carry	C = 0
0111	CY	Carry	C = 1
1100	PO	Parity Odd	P/V = 0
0100	PE	Parity Even	P/V = 1
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
1110	NE	Not Equal	Z = 0
0110	EQ	Equal	Z =
1100	NOV	Overflow is Reset	P/V = u
0100	OV	Overflow is Set	P/V = 1
		Signed Comparisons	
1001	GE	Greater Than or Equal	S ⊕ P/V = 0
0001	LT	Less Than	S ⊕ P/V = 1
1010	GT	Greater Than	Z + (S ⊕ P/V) = 0
0010	LE	Less Than or Equal	Z + (S ⊕ P/V) = 1
		Unsigned Comparisons	
1111	LGE	Logical Greater Than or Equal	C = 0
0111	LLT	Logical Less Than	C = 1
1011	LGT	Logical Greater Than	(C = 0) • (Z = 0) = 1
0011	LLE	Logical Less Than or Equal	C + Z = 1
1000	–	Unconditional (Always True)	–

Notes: • = AND + = OR ⊕ = EXCLUSIVE OR

address called "port address". Conceptually, the port address is very similar to a memory address. Logically, however, port address space is not a part of the memory address space. Although memory and port address information is physically transmitted on the same bus lines in hardware, means are provided to distinguish memory addresses from I/O addresses (using status output lines supplied by the CPU). Port address generation uses the same methodology that is used to generate operand addresses in the nonsegmented CPU using IR and DA addressing modes. In the Instruction Set section these are designated as Port Register (PR) and Port Address (PA) addressing modes.

Two types of I/O instructions are available – "standard I/O" and "special I/O". The address space used by the special I/O is logically separate from the standard I/O. Special I/O address space can be distinguished from the standard I/O space using the status output lines from the CPU. They are intended for communicating with the Memory Management Unit. The I/O instructions exist not only to transfer single words or bytes of data, but also blocks of data from contiguous memory locations.

4.7 INSTRUCTION PREFETCH (PIPELINING)

Most instructions conclude with two or three clock cycles being devoted to internal CPU operations. For such instructions, the subsequent instruction-fetch machine cycle is overlapped with the concluding operations, thereby improving performance by two or three clock cycles per instruction.

Examples of instructions for which the subsequent instruction is fetched while they complete are Arithmetic and Shift instructions.

Some instructions for which the overlap is logically impossible are the Jump instructions (because the following instruction location has not been determined until the instruction completes). Some instructions for which overlap is physically impossible are the Memory Load instructions (because the memory is busy with the current instruction and cannot service the fetch of the succeeding instruction).

4.8 EXTENDED INSTRUCTION PROCESSING

The AmZ8000 architecture has a mechanism for extending the basic instruction set through the use of external devices. Special opcodes have been set aside to implement this feature. When the CPU encounters instructions with these opcodes in its instruction stream, it will perform any indicated address calculation and data transfer, but otherwise treat the "extended instruction" as being executed by the external device. Fields have been set aside in these extended instructions which can be interpreted by external devices (called Extended Processing Units – EPUs) as opcodes. Thus, by using appropriate EPUs, the instruction set of the AmZ8000 can be extended to include specialized instructions.

In general, an EPU is dedicated to performing complex and time consuming tasks in order to unburden the CPU. Typical tasks suitable for specialized EPUs include floating-point arithmetic, data base search and maintenance operations, network interfaces, graphics support operations – a complete list would include most areas of computing. EPUs are generally designed to perform their tasks on data resident in their internal registers. Moving information into and out of the EPU's internal registers, as well as instructing the EPU as to what operations are to be performed, is the responsibility of the CPU.

For the AmZ8000 CPU, control of the EPUs takes the following form. The AmZ8000 CPU fetches instructions, calculates the addresses of operands residing in memory, and controls the movement of data to and from memory. An EPU monitors this activity on the CPU's AD lines. If the instructions fetched by the CPU are extended instructions, all EPUs and the CPU latch the instruction (there may be several different EPUs controlled by one CPU). If the instruction is to be executed by a particular EPU, both the CPU and the indicated EPU will be involved in executing the instruction.

If the extended instruction indicates a transfer of data between the EPU's internal registers and the main memory, the CPU will calculate the memory address and generate the appropriate timing signals ($\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, etc.), but the data transfer itself is between the memory and EPU (over the AD lines). If a transfer of data between the CPU and EPU is indicated, the sender places the data on the AD lines and the receiver reads the AD lines during the next clock period.

If the extended instruction indicates an internal operation to be performed by the EPU, the EPU begins execution of that task and the CPU is free to continue on to the next instrucion. Processing then proceeds simultaneously on both the CPU and the EPU until a second extended instruction is encountered that is destined for the same EPU (if more than one EPU is in the system, all can be operating simultaneously and independently). If an extended instruction specifies an EPU still executing a previous extended instruction, the EPU can suspend instruction fetching by the AmZ8000 CPU until it is ready to accept the next extended instruction: the mechanism for this is the $\overline{STOP}$ line, which suspends CPU activity duing the instruction fetch cycle.

There are four types of extended instructions in the AmZ8000 CPU instruction repertoire; EPU internal operations; data transfers between memory and EPU; data transfers between EPU and CPU; and data transfer between EPU flag registers and CPU flag and control word. The last type is useful when the program must branch based on conditions determined by the EPU. Six opcodes are dedicated to extended instructions: 0E, 0F, 4E, 4F, 8E and 8F (in hexadecimal). The action taken by the CPU upon encountering these instructions is dependent upon an EPU control bit in the CPU's FCW. When this bit is set, it indicates that the system configuration includes EPUs; therefore, the instruction is executed. If this bit is clear, the CPU traps (extended instruction trap),so that a trap handler in software can emulate the desired operation.

In conclusion, the major features of this capability are, that multiple EPUs can be operating in parallel with the CPU, that the five main CPU addressing modes (Register, Immediate, Indirect Register, Direct Address, Indexed) are available in accessing data for the EPU; that each EPU can have more than 256 different instructions; and that data types manipulated by extended instructions can be up to 16 words long.

The extended processing instructions are included in Section 5.8 following the general instruction pages.

5.0 INSTRUCTION SET DETAILS

5.1 INTRODUCTION

This chapter provides detailed descriptions of the AmZ8001 and AmZ8002 instruction set. The instructions are listed by mnemonic in alphabetical order on the following pages.

The reader is referred to previous chapters for descriptions pertaining to items indicated on the detailed instruction pages, such as addressing modes, register designations, instruction format and decoding, flags and condition codes, system and normal instructions, and segmented and nonsegmented modes. The extended instructions discussed in previous sections are listed immediately following the detailed instruction pages.

Information regarding details of the instructions is given in the next sections. This includes a description of the assembler syntax shown used, a key to the instruction pages, and a summary of architectural data for quick reference in understanding these details.

5.2 INSTRUCTION NOTATION AND ENCODING

The details for each general instruction on the following pages begin at the top of the page with the mnemonic and name of the instruction. A generic assembler statement for the instruction is shown. Also, system-only instructions are indicated, and in some cases, a reference to a similar instruction is indicated.

Below this information and to the right is given the operation which the instruction implements and a detailed verbal description of the instruction. Where necessary, a discussion of assembler notation is also given. At the bottom of the page the operation of the flags is given showing which flags are set, cleared, unaffected, or conditionally changed. The conditional changes are defined.

The left section of the page shows each specific form of the general instruction. This includes all available addressing modes and gives the format for both segmented and nonsegmented versions. Both segmented short offset and segmented long offset formats are given. Each specific form indicates the binary machine code with variable binary operand fields defined. Above the machine code representation is shown the specific assembler language syntax for that form. Also shown to the right of the machine code is the number of clocks required for execution of the instruction for that addressing mode.

5.2.1 Instruction Mnemonics

Each instruction page is listed by mnemonic in alphabetical order. Instructions with byte, word, and long word data operands are described on separate pages. The mnemonic suffix “B” refers to a byte instruction, the suffix “L” refers to a long word instruction, and no suffix designates a word instruction. An example is the Shift Dynamic Logical instruction: SDLB, SDLL, and SDL, respectively. For some instructions, a data size either is not applicable or depends on the segmentation mode; here, the mnemonic does not have a suffix to indicate data size.

Some instructions have the relative address (RA) addressing mode. These are indicated with a mnemonic suffix “R.” An example is LOAD Word into Register, LD and LDR. Note that the relative form is included on the general instruction page (LD) as well as being listed on a separate page (LDR).

The letter “R” is also used in the mnemonics to denote a repeat version of an autoindexing instruction. An example is the instruction INPUT Word from I/O Port to Memory, Autoincrement and Repeat (INIR).

5.2.2 Instruction Encoding

The binary encoding of the instruction is given for both segmented (including SSO and SLO where applicable) and nonsegmented versions for each addressing mode. Fields specifying register operands, such as “Rbs,” RRd,” etc. and other operands, such as “n,” are similar to the assembler language syntax description of the instruction. The binary encoding for the register fields is repeated as Table 5.5.1. Some restrictions on register fields are noted in the following sections.

In the case of nonsegmented instructions the designation “ADDRESS” is used to indicate a 16-bit binary address. In the case of segmented mode instructions the address fields are designated as “SEGMENT” and “OFFSET.” In some RA and BA addressing mode instructions “DISPLACEMENT” indicates a binary field containing a displacement supplied by the assembler from the label (address), displacement, or index specified in the assembler language syntax. Other fields specify condition codes (“CC”), number of or location of bits or shift positions (“b” or “n”), or flags (“C”, “Z”, “S”, “PV”, “V”, “N”, etc).

These and other notations are defined in following sections.

Note from Figure 5.5.1 that bit eight of the opcode distinguishes between a word instruction (bit 8 = 1) or a byte instruction (bit 8=0).

Appendix E gives a complete opcode map for the AmZ8001 and AmZ8002 CPUs.

5.2.3 Addressing Mode Encoding

Section 4.3 and Figure 3.6.1 discuss address mode encoding within the instruction opcodes. Pertinent figures are included in the summary material of Section 5.5.

The instruction encodes the addressing modes by using the mode bits (15 and 14), bits 13 and 12, and the register field (bit 7 through bit 4). The specification of general purpose register (pair) zero, R0 or RR0, in the register field of the instruction is used to distinguish between combinations of the addressing modes.

If the instruction mode bits 15 and 14 are “00” and bits 13 and 12 are not “11”, then specifying R0 (or RR0) in the register field (designation = “0000”) denotes IM mode, and any other register designation denotes IR mode. (There are exceptions to this.) Likewise, if the mode bits are “00” and bits 13 and 12 are “11,” then specifying R0 denotes RA mode, and any other register designation denotes BA mode.

If the mode bits are “01,” specifying R0 denotes DA mode, and any other register specification designates X or BX mode. Mode bits “10” designate R addressing mode, and “11” designate special instructions such as the short one-word instructions.

5.2.4 Use of Register R0 and RR0

From the above discussion it is seen that R0 (or RR0) cannot be designated in the instruction register field for the X, BA, and BX addressing modes. That is, general purpose register (pair) zero cannot be used as an index register in X mode or as a base address register in BA or BX modes. In the IR mode some instructions allow the designation of R0 while other instructions do not. The instruction pages following accurately define, in the instruction register fields and descriptions, whether R0 or RR0 can be designated.

Another restriction of using R0 or RR0 is as a stack pointer when using the PUSH and POP instructions. Register (pair) zero cannot be designated as a stack pointer.

To summarize R0 (or RR0) can be designated just as any other register (pair) in the instruction register field with some restrictions. Whether R0 can be designated in a particular addressing mode or form of an instruction is shown on the detailed instruction pages. The general rules for using R0 (and RR0) are listed here.

1. *X Mode:* R0 cannot be designated as the index register in indexed (X) addressing mode.
2. *BA, BX Modes:* R0 cannot be designated as the base address register in base address (BA) or base indexed (BX) addressing modes. Note that R0 or RR0 can be designated as the index register in the BX mode.
3. *IR Mode:* R0 cannot be designated as the indirect register in the IR mode if that instruction has the immediate form (IM mode) also available. Two other considerations apply for the IR mode:
   a. If no IM mode of the instruction is available, R0 (or RR0) can be designated as the indirect register in the IR mode. An example of this is the INCREMENT Word Instruction, INC, for both segmented and nonsegmented versions of the IR addressing mode.
   b. In other cases where the IM mode of the instruction is not available, the designation of R0 (or RR0) is not allowed in that it is used to distinguish a different version or instruction. An example is the SET Bit in Word instruction, SET, which has no IM mode. Specifying R0 or RR0 in IR mode of the static form of the instruction is not allowed because specifying "0000" in the register field is the opcode for the dynamic form of the instruction (R mode).
4. *Stack Pointer:* R0 or RR0 cannot be designated as stack pointers in the PUSH and POP instructions.

5.3 ASSEMBLER LANGUAGE SYNTAX

Each form of an instruction (including byte, word, and long word, segmented and nonsegmented versions, and each available addressing mode) is shown with its corresponding assembler language statement. The statements follow the general instruction notation used by the MACRO8000 assembler. For additional information refer to the MACRO8000 Assembler User's Manual.

The assembler syntax is given for each mnemonic at the top center of each page just below the name of the instruction. This statement is a single generic form that covers all variations of the instruction on that page. Though generic, the statement shows specific operands wherever possible; these are indicated by upper case letters. Lower case letters are used to denote variables in the statement for which suitable values are to be substituted. One example of this is "Rs," denoting a word register (R0, R1, ..., or R15) required by the addressing mode as a source register. Another example is the use of "dst" ("src"), indicating a general destination (source) which type is determined by the addressing mode.

The assembler syntax is also given for each specific version of the instruction and is shown directly above its binary machine code representation. Because these are specific forms, the general "dst" and "src" operands are substituted by their specific value for that addressing mode.

Some variables are shown in upper case. These include labels, displacements, lists, and binary and integer values. They are listed in the following section entitled Notation key.

5.4 ASSEMBLER EXCEPTIONS

MACRO8000 does not allow designation of R0 or RR0 as an indirect register in the IR addressing mode. Some instructions do allow the designation of R0 (or RR0) in the AmZ8001 and AmZ8002. These are noted in the detailed instruction pages.

The assembler statements are terminated with a semicolon (;). These have not been shown.

The MACRO8000 does not use the LDA or LDAR mnemonic to implement the LOAD Address instruction. This instruction is implemented using the LD mnemonic with the address specified as an immediate operand address constant. This may generate an LD or LDA instruction object code, depending on the addressing mode and the version of the assembler being used. To specify an address constant to be used as an immediate operand, place a "↑" character in front of the operand. (Refer to comments made in Section 5.6.8 regarding use of the circumflex character, "↑.")

The LDB load byte into a register with an immediate value instruction has two opcodes which will perform the desired operation. MACRO8000 generates the faster and shorter version; other assemblers may support both forms.

The instructions listed below have a comment in the Assembler Notation section as follows: "A LAB or D which results in a displacement outside the allowable range produces an assembler error." Because the displacement range of these instructions in the RA and BA addressing modes is +32,767 to − 32,768, the MACRO8000 assembler can compute the correct displacement if outside the indicated range. It does this by generating a module 65,536 two's complement displacement. For example, a LAB indicating a displacement of −65,520 will produce a positive displacement of +16 by wrapping around.

No error occurs in these cases. Instructions which have displacements assembled this way include:

LD, LDR word register into memory
LD, LDR word into register
LDB, LDRB byte register into memory
LDB, LDRB byte into register
LDL, LDRL long word register into memory
LDL, LDRL long word into register
LDA, LDAR address into register

5.5 SUMMARY OF ARCHITECTURAL DETAILS

The following figures and tables are a reproduction of previous information provided here for easy reference in understanding the instruction set details.

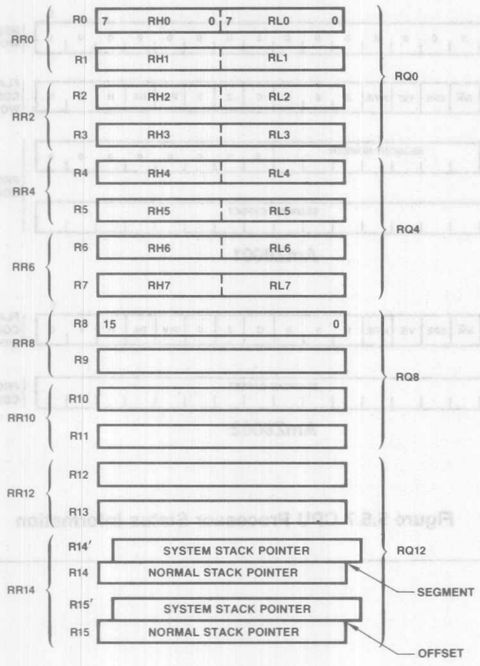


Figure 5.5.1 AmZ8001 General Registers

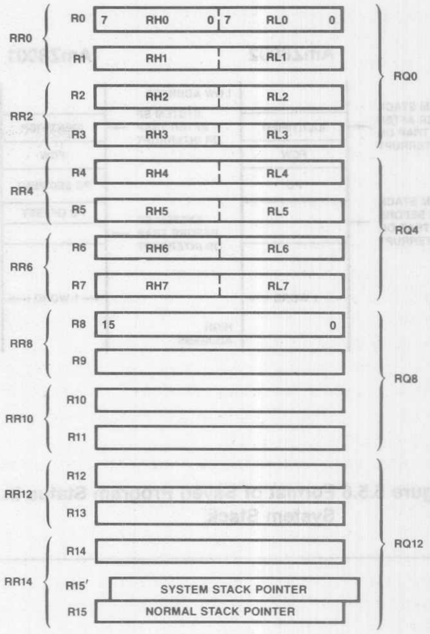


Figure 5.5.2 AmZ8002 General Registers

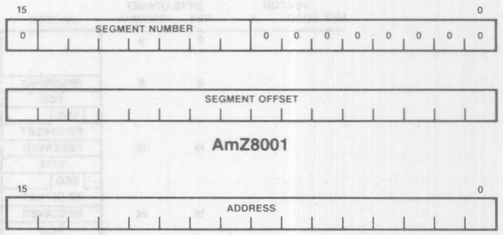
TABLE 5.5.1 GENERAL REGISTER ORGANIZATION AND DESIGNATORS

Register Designator	Byte Mode	Word Mode	Long Word Mode	Quadruple Word Mode
0000	RH0	R0	RR0	RQ0
0001	RH1	R1	–	–
0010	RH2	R2	RR2	–
0011	RH3	R3	–	–
0100	RH4	R4	RR4	RQ4
0101	RH5	R5	–	–
0110	RH6	R6	RR6	–
0111	RH7	R7	–	–
1000	RL0	R8	RR8	RQ8
1001	RL1	R9	–	–
1010	RL2	R10	RR10	–
1011	RL3	R11	–	–
1100	RL4	R12	RR12	RQ12
1101	RL5	R13	–	–
1110	RL6	R14	RR14	–
1111	RL7	R15	–	–

(–Reserved)

Notes:
All general purpose registers can be used as accumulators. However, R0 in the AmZ8002 (and RR0 in the AmZ8001) cannot be used as an index register or memory pointer. Refer to the section on Address Modes (3.6) Section 5.2.4.

The highest order general-purpose registers are used as implied stack pointers. For a description of this refer to the section entitled Stack Pointers (2.3.4).



AmZ8001

AmZ8002

Figure 5.5.3 CPU Program Counters

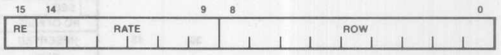
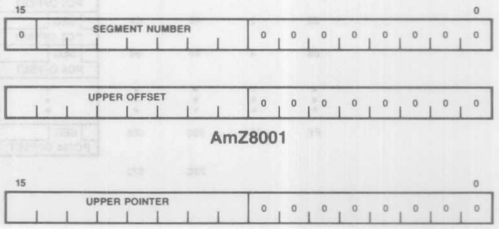


Figure 5.5.4 CPU Refresh Counter



AmZ8001

AmZ8002

Figure 5.5.5 New Program Status Area Pointer

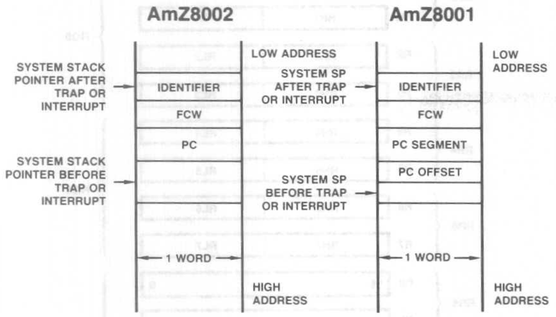

AmZ8002
AmZ8001
LOW ADDRESS
SYSTEM STACK POINTER AFTER TRAP OR INTERRUPT
IDENTIFIER
FCW
PC
SYSTEM STACK POINTER BEFORE TRAP OR INTERRUPT
SYSTEM SP AFTER TRAP OR INTERRUPT
SYSTEM SP BEFORE TRAP OR INTERRUPT
1 WORD
HIGH ADDRESS
LOW ADDRESS
IDENTIFIER
FCW
PC SEGMENT
PC OFFSET
1 WORD
HIGH ADDRESS

Figure 5.5.6 Format of Saved Program Status in the System Stack

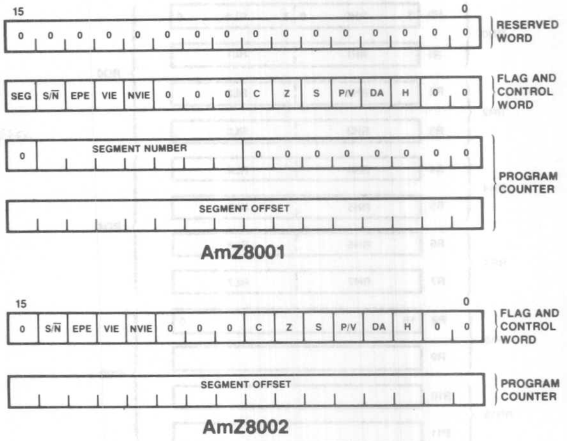


Figure 5.5.7 CPU Processor Status Information

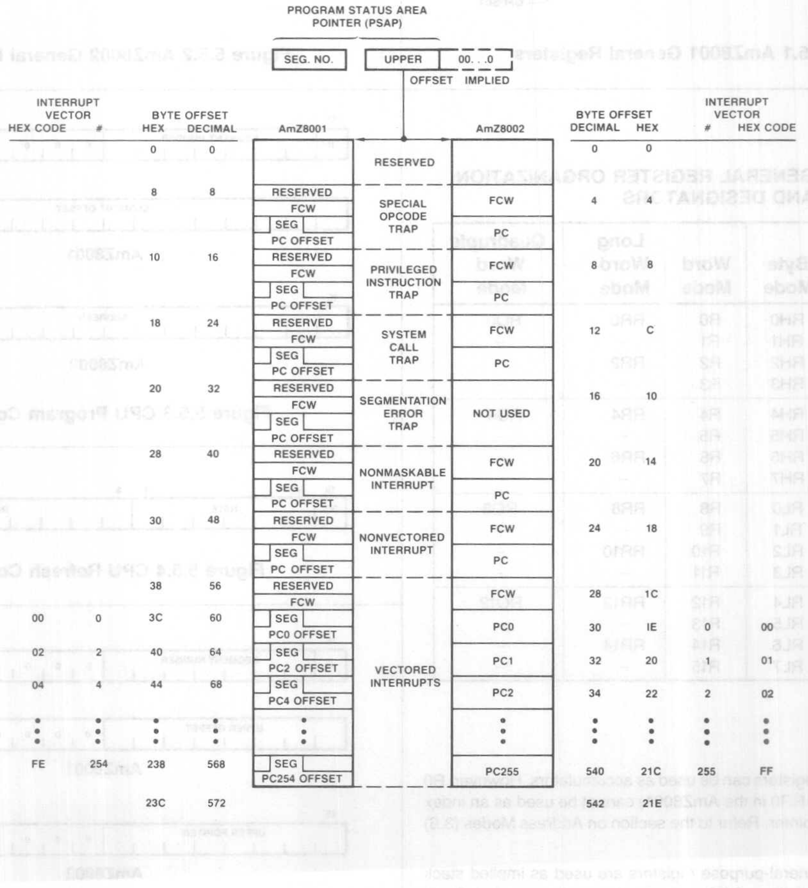


Figure 5.5.8 Program Status Area

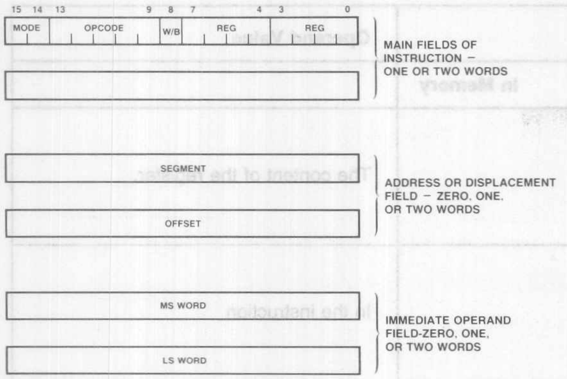

15 14 13 9 8 7 4 3 0
MODE
OPCODE
W/B
REG
REG
MAIN FIELDS OF INSTRUCTION – ONE OR TWO WORDS
SEGMENT
OFFSET
ADDRESS OR DISPLACEMENT FIELD – ZERO, ONE, OR TWO WORDS
MS WORD
LS WORD
IMMEDIATE OPERAND FIELD-ZERO, ONE, OR TWO WORDS

Figure 5.5.9 General Instruction Word Format

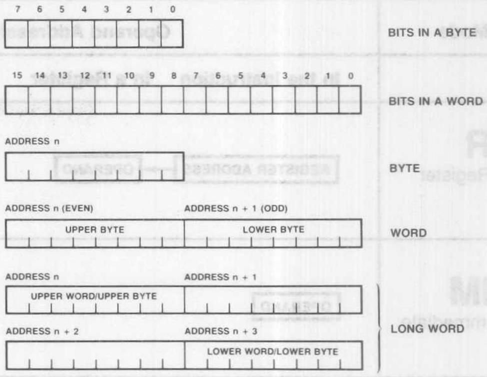


Figure 5.5.10 Addressable Data Elements

TABLE 5.2 INSTRUCTION DECODING

Address Mode

Bits 15, 14	Bits 13, 12	Bits 7, 6, 5, 4	Mode
00	Not 11	0	IM
00	Not 11	Not 0	IR
00	11	0	RA
00	11	Not 0	BA
01	X	0	DA
01	Not 11	Not 0	X
01	11	Not 0	BX
10	X	X	R
11	X	X	Special*

*Used for short one-word instructions.

Word/Byte

Bit 8	Mode
0	Byte
1	Word

Address, When Present

Displacement -	16-Bit Word
Nonsegmented Address	16-Bit Word
Segmented Short Offset	0 + 7-Bit Segment + 8-Bit Offset
Segmented Long Offset	First Word: 1 + 7-Bit Segment + 8-Bit Unused Second Word: 16-Bit Offset

Immediate Data, When Present

Byte	Same Byte in Both Halves of Word
Word	16-Bit Word
Long-Word	First Word: Bits 16:31 of Operand Second Word: Bits 0:15 of Operand

TABLE 5.3 CC-FIELD DECODING

CC Field	Assembler Notation	Meaning	Flag Settings for CC True
1110	NZ	Not Zero	Z = 0
0110	ZR	Zero	Z = 1
1111	NC	No Carry	C = 0
0111	CY	Carry	C = 1
1100	PO	Parity Odd	P/V = 0
0100	PE	Parity Even	P/V = 1
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
1110	NE	Not Equal	Z = 0
0110	EQ	Equal	Z = 1
1100	NOV	Overflow is Reset	P/V = 0
0100	OV	Overflow is Set	P/V = 1
1001	GE	Greater Than or Equal	S ⊕ P/V = 0
0001	LT	Less Than	S ⊕ P/V = 1
1010	GT	Greater Than	Z + (S ⊕ P/V) = 0
0010	LE	Less Than or Equal	Z + (S ⊕ P/V) = 1
1111	LGE	Logical Greater Than or Equal	C = 0
0111	LLT	Logical Less Than	C = 1
1011	LGT	Logical Greater Than	(C = 0) • (Z = 0) = 1
0011	LLE	Logical Less Than or Equal	C + Z = 1
1000	–	Unconditional (Always True)	–

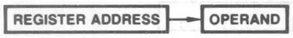
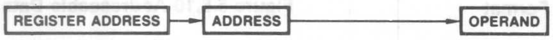
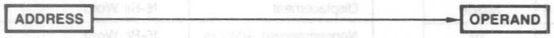
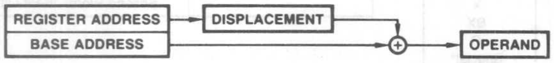
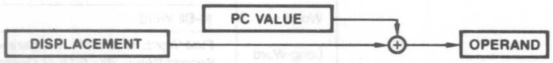
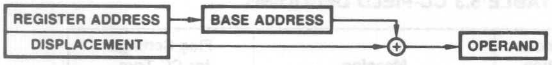
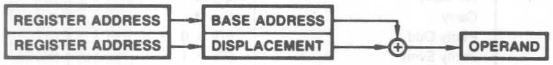
Mode	Operand Addressing	Operand Value
	In the Instruction In a Register In Memory	
R Register		The content of the register.
IM Immediate		In the instruction
IR Indirect Register		The content of the location whose address is in the register.
DA Direct Address		The content of the location whose address is in the instruction.
X Index		The content of the location whose address is the address in the instruction, offset by the content of the working register.
RA Relative Address		The content of the location whose address is the content of the program counter, offset by the displacement in the instruction.
BA Base Address		The content of the location whose address is the address in the register, offset by the displacement in the instruction.
BX BX Base Index	 REGISTER ADDRESS REGISTER ADDRESS BASE ADDRESS DISPLACEMENT OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the register.

Figure 5.5.11 Addressing Modes

5.6 NOTATION KEY

Abbreviations and notations used on the detailed instruction pages are included in this section.

5.6.1 Appended Information

Refer to the appendices for additional information:

Appendix A	AmZ8000 Instruction Set: By Logical Group
Appendix B	AmZ8000 Instruction Set: Numeric Listing by Opcode
Appendix C	AmZ8000 Instruction Set: Alphabetic Listing By Mnemonic
Appendix D	AmZ8000 Instruction Set: Topical Index
Appendix E	AmZ8000 Instruction Set: Opcode Map
Appendix F	Executive Module Sample Code
Appendix G	ASCII Character Set
Appendix H	Powers of 2 and 16
Appendix I	Hexadecimal and Decimal Integer Conversion Table

5.6.2 Addressing Mode and Segmentation Notation

Notation used to designate the addressing modes is shown and includes examples of assembler syntax operands.

Notation	Addressing Mode	Assembler Syntax Example
R	Register	R7
IM	Immediate	4 CONSTANT ↑LABEL
IR	Indirect Register	R7↑
DA	Direct Address	LABEL #4D8B LABEL + DISPLACEMENT
X	Index	LABEL (R7) #4D8B↑ (R7)
RA	Relative Address	LAB ↑$ + #100)
BA	Base Address	R7↑(DISPLACEMENT) R7↑(#100)
BX	Base Index	R7↑(R1)
PR	Port Register	R7
PA	Port Address	4 CONSTANT

The segmentation version notation is:

NS –	nonsegmented
S –	segmented
SSO –	segmented short offset
SLO –	segmented long offset

5.6.3 Source and Destination Notation

Operand sources and destinations are indicated in the general instruction assembler statement at the top of the page. Non-specific forms are indicated by lower case "src" and "dst." This implies that the addressing mode determines the specific source or destination.

Specific sources and destinations are listed for each form and addressing mode of the instruction and, if all versions on the page are identical, at the top of the page. The specific source is designated by a lower case "s" suffix behind a register notation. The specific destination is designated by a lower case "d" suffix behind a register notation. Examples of these are "Rs" and "RRd."

5.6.4 Register Notation

Notation	Meaning	Range/Examples	Comments
*R	word register	R0, R1,..., R15	16 bits
*Rb	byte register	RL0, RH0, RL1,..., RH7	8 bits
*RR	register pair	RR0, RR2,..., RR14	32 bits
*RQ	register quad	RQ0, RQ4, RQ8, RQ12	64 bits
Rx	index register	as, R7	word register only
Rc	counter register	as, R10	word register only
Rbc	byte counter register	as, RH7	byte register only
Rp	port register	as, R14	word register only

* – Appended with a source or destination indication, "s" or "d." Examples are Rs, Rbd, RRs, RQd, etc.

5.6.5 Operand Notation

In addition to register operand designations, notations used for operands and other values are listed here both for the assembler language statements and the instruction opcodes. Upper case is used for statement operands and correspond to an instruction field value, typically indicated in lower case. The assembler assembles the statement operand into the appropriate binary code. A statement about assembler notation is given on instruction pages where appropriate and includes information on range and relationship between assembler and opcode values.

Notation	Description
b, B	bit number or number of bit positions
c, C	count or carry
CC	condition code
CR	control register
d, D	signed, 2's complement displacement
h	hex integer; 01, 1, ..., F
IM	immediate operand (IM = word, IMb = byte, IMℓ= long-word, IMd = digit)
LIST	list of optional (all, any or none) values
n, N	decimal integer

5.6.6 Address and Label Notation

The general assembler statements and operations use "addr" to denote a memory address. These addresses are generated by the assembler depending on the addressing mode and assembler syntax, such as labels, displacements, and indexes. Addresses also take the form #hh (segment) or #hhhh (offset). An example is #4D8B, the # symbol designating a hex integer value.

Notation	Description
ADDRESS	Denotes a 16-bit binary address field in the instruction generated by the assembler for nonsegmented versions of DA and X mode instructions.
DISPLACEMENT	A signed or unsigned 2's complement integer used in the RA and BA addressing modes and relative instructions such as CALL Subroutine (CALR), JUMP Conditional (JR), and LOAD instructions such as LDR. The instruction description and assembler notation define whether the displacement is added or subtracted and what the range is.
LAB	An assembler syntax label designating the relative address (RA) mode. The assembler uses this label to generate the instruction displacement relative to the updated PC.
LABEL	An absolute address in memory specified to the assembler either as a label or as an absolute address of the form #hhhh.
LABSSO	A label designating a segmented short offset (SSO) form of address. This is composed of a 7-bit segment address and the lowest byte of a 16-bit offset address. It is used to denote the SSO version of DA and X mode instructions.
OFFSET	A field used to specify the offset address (16 bits for SLO or 8 bits for SSO) in a segmented form of DA or X mode instructions.
PORT	An assembler syntax used to denote a port address (a 1-bit field) in a port address (PA) mode of the (special) I/O instructions.
SEGMENT	A field used to specify the 7-bit segment address in segmented (SSO or SLO) forms of DA and X mode instructions.

Note that the assembler uses segment directives, not instruction statements, to specify segment address to be used during assembly. Refer to the assembler reference manual.

5.6.7 Condition Codes and Other Notations

The condition codes use a notation defined in Table. Also shown are the CPU flag settings. In addition to these, other notations are used:

Notation	Meaning	Instruction Example
CC	condition code	JR
CR	control register	LDCTL
FCW	flag and control word	LDCTL
FLAGS	flag byte of FCW	LDCTLB
N	non-vectored interrupt opcode bit	EI
NSPOFF	normal stack pointer offset	LDCTL
NSPSEG	normal stack pointer segment	LDCTL
NVIE	non-vectored interrupt flag	EI
PSAPOFF	NPSAP upper offset	LDCTL
PSAPSEG	NPSAP segment	LDCTL
REFRESH	refresh register	LDCTL
SGN	sign bit	COMFIG
V	vectored interrupt opcode bit	EI
VIE	vectored interrupt flag	EI

5.6.8 Special Character Notation

The operations described on each instruction page use the following convention with respect to special characters:

Symbol	Description
−	"is replaced by"
< >	denotes bit field, range or position
( )	denotes "contents of"
:	indicates lowest and highest bit positions
;	delimiter for list of optional entries
+	add
−	subtract
x	times
÷ or /	divide
V	logical OR
Λ	logical AND
+	logical EXCLUSIVE OR
−	logical complement
→---	denotes direction of rotation or shift
⟷	exchange
=	equals
≠	not equals

The following list includes notations used by the assembler and assembler syntax:

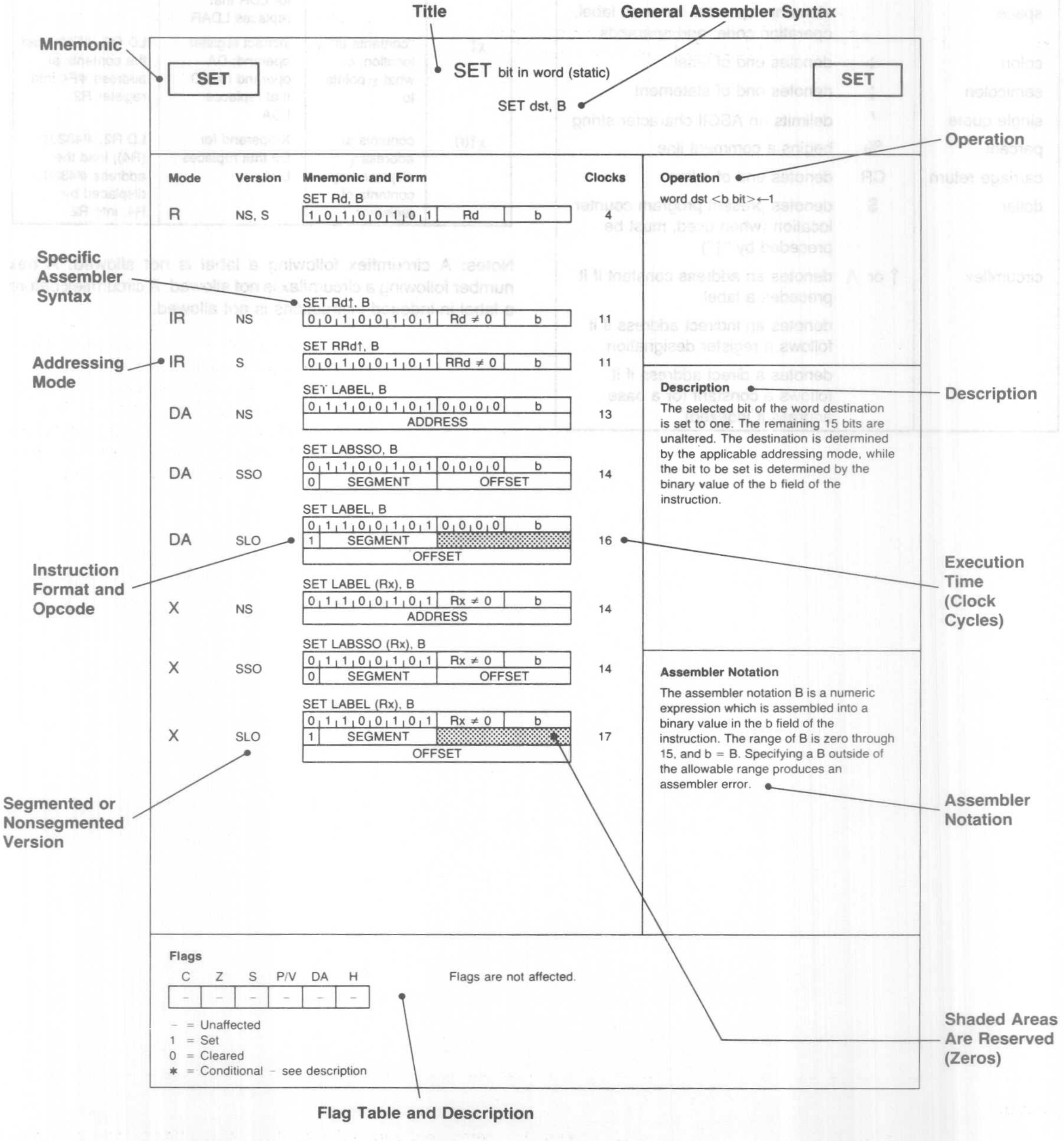
Symbol	Description	
number	#	denotes a hex constant
parentheses	( )	enclose a subscript index register
comma	,	separates multiple operands
space		element separator between label, operation code, and operands
colon	:	denotes end of label
semicolon	;	denotes end of statement
single quote	′	delimits an ASCII character string
percent	%	begins a comment line
carriage return	CR	denotes end of a line
dollar	$	denotes present program counter location (when used, must be preceded by "↑")
circumflex	↑ or ∧	denotes an address constant if it precedes a label
		denotes an indirect address if it follows a register designation
		denotes a direct address if it follows a constant (or a base address if indexed)

The circumflex symbol is used in several ways, including as LDA (and LDAR) LOAD ADDRESS (Relative) equivalents, and is summarized as follows:

Notation	Meaning	Use	Example
↑χ	"address of" χ, or pointer to χ	DA operand for LD that replaces LDA; RA operand for LDR that replaces LDAR	LD R2, ↑L3; load the address of L3 into register R2
χ↑	"contents at" χ location, or what χ points to	indirect register operand; DA operand for LD that replaces LDA	LD R2, #F4↑; load the contents at address #F4 into register R2
χ↑(r)	contents at address χ displaced by contents of register r	X operand for LD that replaces LDA	LD R2, #4320↑ (R4); load the address #4320, displaced by R4, into R2

Notes: A circumflex following a label is not allowed. A hex number following a circumflex is not allowed. A circumflex before a label in indexed instructions is not allowed.

5.6.9 INSTRUCTION PAGE KEY


Title
General Assembler Syntax
Mnemonic
Operation
Specific Assembler Syntax
Addressing Mode
Description
Instruction Format and Opcode
Execution Time (Clock Cycles)
Segmented or Nonsegmented Version
Assembler Notation
Shaded Areas Are Reserved (Zeros)
Flag Table and Description

SET

SET bit in word (static)

SET dst, B

SET

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SET Rd, B 1 0 1 0 0 1 0 1 | Rd | b	4
IR	NS	SET Rd↑, B 0 0 1 0 0 1 0 1 | Rd ≠ 0 | b	11
IR	S	SET RRd↑, B 0 0 1 0 0 1 0 1 | RRd ≠ 0 | b	11
DA	NS	SET LABEL, B 0 1 1 0 0 1 0 1 | 0 0 0 0 | b ADDRESS	13
DA	SSO	SET LABSSO, B 0 1 1 0 0 1 0 1 | 0 0 0 0 | b 0 | SEGMENT | OFFSET	14
DA	SLO	SET LABEL, B 0 1 1 0 0 1 0 1 | 0 0 0 0 | b 1 | SEGMENT | ▨ OFFSET	16
X	NS	SET LABEL (Rx), B 0 1 1 0 0 1 0 1 | Rx ≠ 0 | b ADDRESS	14
X	SSO	SET LABSSO (Rx), B 0 1 1 0 0 1 0 1 | Rx ≠ 0 | b 0 | SEGMENT | OFFSET	14
X	SLO	SET LABEL (Rx), B 0 1 1 0 0 1 0 1 | Rx ≠ 0 | b 1 | SEGMENT | ▨ OFFSET	17

Operation

word dst <b bit>←1

Description

The selected bit of the word destination is set to one. The remaining 15 bits are unaltered. The destination is determined by the applicable addressing mode, while the bit to be set is determined by the binary value of the b field of the instruction.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

ADC

ADD words with carry

ADC

ADC Rd, Rs

Mode	Version	Mnemonic and Form			Clocks
		ADC Rd, Rs			
R	NS, S	1 0 1 1 0 1 0 1	Rs	Rd	5

Operation

Rd<0:15>←Rs<0:15>+Rd<0:15>+C

Description

The contents of the general-purpose registers designated by the Rs (source) and Rd (destination) fields of the instruction are added together along with the carry flag to obtain the result. The 16-bit result is loaded into the destination register, whose original contents are lost. The contents of the source are not altered.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if there is carry from the most significant bit position of the word. Reset otherwise.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

ADCB

ADD byte with carry

ADCB

ADCB Rbd, Rbs

Mode	Version	Mnemonic and Form	Clocks
		ADCB Rbd, Rbs	
R	NS, S	1 0 1 1 0 1 0 0 | Rbs | Rbd	5

Operation

Rbd<0:7>←Rbs<0:7>+Rbd<0:7>+C

Description

The contents of the general-purpose byte registers designated by the Rbs (source) and Rbd (destination) fields of the instruction are added together along with the carry flag to obtain the result. The 8-bit result is loaded into the destination register, whose original contents are lost. The contents of the source are not altered.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	0	*

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if there is a carry from the most significant bit position of the byte. Reset otherwise.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.
DA: Reset always.
H: Set to 1 on carry from the least significant digit of result. Reset otherwise.

ADD ADD word to register **ADD**

ADD Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	ADD Rd, Rs 1 0 0 0 0 0 0 1 | Rs | Rd	4
IM	NS, S	ADD Rd, IM 0 0 0 0 0 0 0 1 | 0 0 0 0 | Rd OPERAND	7
IR	NS	ADD Rd, Rs↑ 0 0 0 0 0 0 0 1 | Rs ≠ 0 | Rd	7
IR	S	ADD Rd, RRs↑ 0 0 0 0 0 0 0 1 | RRs ≠ 0 | Rd	7
DA	NS	ADD Rd, LABEL 0 1 0 0 0 0 0 1 | 0 0 0 0 | Rd ADDRESS	9
DA	SSO	ADD Rd, LABSSO 0 1 0 0 0 0 0 1 | 0 0 0 0 | Rd 0 | SEGMENT | OFFSET	10
DA	SLO	ADD Rd, LABEL 0 1 0 0 0 0 0 1 | 0 0 0 0 | Rd 1 | SEGMENT | ▨ OFFSET	12
X	NS	ADD Rd, LABEL (Rx) 0 1 0 0 0 0 0 1 | Rx ≠ 0 | Rd ADDRESS	10
X	SSO	ADD Rd, LABSSO (Rx) 0 1 0 0 0 0 0 1 | Rx ≠ 0 | Rd 0 | SEGMENT | OFFSET	10
X	SLO	ADD Rd, LABEL (Rx) 0 1 0 0 0 0 0 1 | Rx ≠ 0 | Rd 1 | SEGMENT | ▨ OFFSET	13

Operation

Rd<0:15>←src<0:15>+Rd<0:15>

Description

Source operand and destination operand words are added together and the 16-bit result is loaded into the destination. The contents of the source are not altered and the original contents of the destination are lost. The source is determined by the applicable addressing mode and the destination is always a general-purpose register designated by the Rd field of the instruction.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Set to 1 if there is a carry from the most significant bit position of the word. Reset otherwise.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

ADDB

ADD byte to register

ADDB

ADDB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	ADDB Rbd, Rbs 1 0 0 0 0 0 0 0 | Rbs | Rbd	4
IM	NS, S	ADDB Rbd, IMb 0 0 0 0 0 0 0 0 | 0 0 0 0 | Rbd 7 OPERAND 0 | 7 OPERAND 0	7
IR	NS	ADDB Rbd, Rs↑ 0 0 0 0 0 0 0 0 | Rs ≠ 0 | Rbd	7
IR	S	ADDB Rbd, RRs↑ 0 0 0 0 0 0 0 0 | RRs ≠ 0 | Rbd	7
DA	NS	ADDB Rbd, LABEL 0 1 0 0 0 0 0 0 | 0 0 0 0 | Rbd ADDRESS	9
DA	SSO	ADDB Rbd, LABSSO 0 1 0 0 0 0 0 0 | 0 0 0 0 | Rbd 0 | SEGMENT | OFFSET	10
DA	SLO	ADDB Rbd, LABEL 0 1 0 0 0 0 0 0 | 0 0 0 0 | Rbd 1 | SEGMENT | ▨ OFFSET	12
X	NS	ADDB Rbd, LABEL (Rx) 0 1 0 0 0 0 0 0 | Rx ≠ 0 | Rbd ADDRESS	10
X	SSO	ADDB Rbd, LABSSO (Rx) 0 1 0 0 0 0 0 0 | Rx ≠ 0 | Rbd 0 | SEGMENT | OFFSET	10
X	SLO	ADDB Rbd, LABEL (Rx) 0 1 0 0 0 0 0 0 | Rx ≠ 0 | Rbd 1 | SEGMENT | ▨ OFFSET	13

Operation

Rbd<0:7>←src<0:7>+Rbd<0:7>

Description

Source operand and destination operand bytes are added together and the 8-bit result is loaded into the destination. The contents of the source are not altered and the original contents of the destination are lost. The source is determined by the applicable addressing mode and the destination is always a general-purpose byte register designated by the Rbd field of the instruction.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	0	*

– = *Unaffected*
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if there is a carry from the most significant bit position of the byte. Reset otherwise.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.
DA: Always reset.
H: Set to 1 if there is a carry from the least significant digit. Reset otherwise.

ADD long word to register

ADDL RRd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	ADDL RRd, RRs 1 0 0 1 0 1 1 0 RRs RRd	8
IM	NS, S	ADDL RRd, IMℓ 0 0 0 1 0 1 1 0 0 0 0 0 RRd 31 OPERAND 16 15 OPERAND 0	14
IR	NS	ADDL RRd, Rs↑ 0 0 0 1 0 1 1 0 Rs ≠ 0 RRd	14
IR	S	ADDL RRd, RRs↑ 0 0 0 1 0 1 1 0 RRs ≠ 0 RRd	14
DA	NS	ADDL RRd, LABEL 0 1 0 1 0 1 1 0 0 0 0 0 RRd ADDRESS	15
DA	SSO	ADDL RRd, LABSSO 0 1 0 1 0 1 1 0 0 0 0 0 RRd 0 SEGMENT OFFSET	16
DA	SLO	ADDL RRd, LABEL 0 1 0 1 0 1 1 0 0 0 0 0 RRd 1 SEGMENT ▨ OFFSET	18
X	NS	ADDL RRd, LABEL (Rx) 0 1 0 1 0 1 1 0 Rx ≠ 0 RRd ADDRESS	16
X	SSO	ADDL RRd, LABSSO (Rx) 0 1 0 1 0 1 1 0 Rx ≠ 0 RRd 0 SEGMENT OFFSET	16
X	SLO	ADDL RRd, LABEL (Rx) 0 1 0 1 0 1 1 0 Rx ≠ 0 RRd 1 SEGMENT ▨ OFFSET	19

Operation

RRd<0:31>←src<0:31>+RRd<0:31>

Description

Source operand and destination operand long words are added together and the result is loaded into the destination. The contents of the source are not altered and the original contents of the destination are lost. The source is determined by the applicable addressing mode and the destination is always a general-purpose register pair designated by the RRd field of the instruction.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if there is a carry from the most significant bit position of the long word. Reset otherwise.

Z: Set to 1 if result is zero. Reset otherwise.

S: Set to 1 if result is negative. Reset otherwise.

P/V: Set to 1 on arithmetic overflow. Reset otherwise.

AND word with register

AND Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	AND Rd, Rs 1 0 0 0 0 1 1 1 Rs Rd	4
IM	NS, S	AND Rd, IM 0 0 0 0 0 1 1 1 0 0 0 0 Rd OPERAND	7
IR	NS	AND Rd, Rs↑ 0 0 0 0 0 1 1 1 Rs ≠ 0 Rd	7
IR	S	AND Rd, RRs↑ 0 0 0 0 0 1 1 1 RRs ≠ 0 Rd	7
DA	NS	AND Rd, LABEL 0 1 0 0 0 1 1 1 0 0 0 0 Rd ADDRESS	9
DA	SSO	AND Rd, LABSSO 0 1 0 0 0 1 1 1 0 0 0 0 Rd 0 SEGMENT OFFSET	10
DA	SLO	AND Rd, LABEL 0 1 0 0 0 1 1 1 0 0 0 0 Rd 1 SEGMENT ▨ OFFSET	12
X	NS	AND Rd, LABEL (Rx) 0 1 0 0 0 1 1 1 Rx ≠ 0 Rd ADDRESS	10
X	SSO	AND Rd, LABSSO (Rx) 0 1 0 0 0 1 1 1 Rx ≠ 0 Rd 0 SEGMENT OFFSET	10
X	SLO	AND Rd, LABEL (Rx) 0 1 0 0 0 1 1 1 Rx ≠ 0 Rd 1 SEGMENT ▨ OFFSET	13

Operation

Rd<0:15>←Rd<0:15> ∧ src<0:15>

Description

A logical AND operation is performed between the corresponding bits of the source and destination words. The source operand is determined by the applicable addressing mode, while the destination operand is always a general-purpose word register, designated by the Rd field of the instruction. The result of the operation is loaded into the destination, whose original contents are lost. The source contents are not altered.

Flags

C	Z	S	P/V	DA	H
–	∗	∗	–	–	–

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

AND byte with register

ANDB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	ANDB Rbd, Rbs 1 0 0 0 0 1 1 0 Rbs Rbd	4
IM	NS, S	ANDB Rbd, IMb 0 0 0 0 0 1 1 0 0 0 0 0 Rbd 7 OPERAND 0 7 OPERAND 0	7
IR	NS	ANDB Rbd, Rs↑ 0 0 0 0 0 1 1 0 Rs ≠ 0 Rbd	7
IR	S	ANDB Rbd, RRs↑ 0 0 0 0 0 1 1 0 RRs ≠ 0 Rbd	7
DA	NS	ANDB Rbd, LABEL 0 1 0 0 0 1 1 0 0 0 0 0 Rbd ADDRESS	9
DA	SSO	ANDB Rbd, LABSSO 0 1 0 0 0 1 1 0 0 0 0 0 Rbd 0 SEGMENT OFFSET	10
DA	SLO	ANDB Rbd, LABEL 0 1 0 0 0 1 1 0 0 0 0 0 Rbd 1 SEGMENT ▨ OFFSET	12
X	NS	ANDB Rbd, LABEL (Rx) 0 1 0 0 0 1 1 0 Rx ≠ 0 Rbd ADDRESS	10
X	SSO	ANDB Rbd, LABSSO (Rx) 0 1 0 0 0 1 1 0 Rx ≠ 0 Rbd 0 SEGMENT OFFSET	10
X	SLO	ANDB Rbd, LABEL (Rx) 0 1 0 0 0 1 1 0 Rx ≠ 0 Rbd 1 SEGMENT ▨ OFFSET	13

Operation

Rbd<0:7>←Rbd<0:7> ∧ src<0:7>

Description

A logical AND operation is performed between the corresponding bits of the source and destination bytes. The source operand is determined by the applicable addressing mode, while the destination operand is always a general-purpose byte register, designated by the Rbd field of the instruction. The result of the operation is loaded into the destination, whose original contents are lost. The source contents are not altered.

Flags

C	Z	S	P/V	DA	H
–	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if parity of result is even. Reset otherwise.

BIT

BIT test in a word (dynamic)

BIT Rd, Rs

BIT

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	BIT Rd, Rs	10

0 0 1 0	0 1 1 1	0 0 0 0	Rs
▨	Rd	▨	▨

Operation

Z flag←Rd<bit specified in Rs(0:3)>

Description

The selected bit of the word destination register is tested and the Z flag is affected. The destination word operand is the general-purpose register designated by the Rd field of the instruction. The bit to be tested is determined from a binary decode of the least significant four bits of a general-purpose word register designated by the Rs field. The contents of the destination are unaltered.

Flags

C	Z	S	P/V	DA	H
–	*	–	–	–	–

Z: Set to 1 if selected bit of destination operand is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

BIT

BIT test in a word (static)

BIT

BIT dst, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	BIT Rd, B 1 0 1 0 0 1 1 1 | Rd | b	4
IR	NS	BIT Rd↑, B 0 0 1 0 0 1 1 1 | Rd ≠ 0 | b	8
IR	S	BIT RRd↑, B 0 0 1 0 0 1 1 1 | RRd ≠ 0 | b	8
DA	NS	BIT LABEL, B 0 1 1 0 0 1 1 1 | 0 0 0 0 | b ADDRESS	10
DA	SSO	BIT LABSSO, B 0 1 1 0 0 1 1 1 | 0 0 0 0 | b 0 | SEGMENT | OFFSET	11
DA	SLO	BIT LABEL, B 0 1 1 0 0 1 1 1 | 0 0 0 0 | b 1 | SEGMENT | ▨ OFFSET	13
X	NS	BIT LABEL (Rx), B 0 1 1 0 0 1 1 1 | Rx ≠ 0 | b ADDRESS	11
X	SSO	BIT LABSSO (Rx), B 0 1 1 0 0 1 1 1 | Rx ≠ 0 | b 0 | SEGMENT | OFFSET	11
X	SLO	BIT LABEL (Rx), B 0 1 1 0 0 1 1 1 | Rx ≠ 0 | b 1 | SEGMENT | ▨ OFFSET	14

Operation

Z flag ← $\overline{\text{word dst<b bit>}}$

Description

A bit in the word destination is tested, and the Z flag is affected as shown below.The destination is determined by the applicable addressing mode, while the bit to be tested is specified by the binary value of the b field of the instruction. The contents of the destination are not altered.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of b is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	∗	–	–	–	–

Z: Set to 1 if specified bit of destination word is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

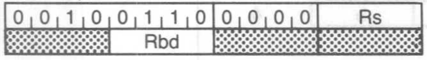
BITB

BIT test in a byte (dynamic)

BITB

BIT Rbd, Rs

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	BIT Rbd, Rs	10



Operation

Z flag←$\overline{\text{Rbd<bit specified in Rs(0:2)>}}$

Description

The selected bit of the byte destination register is tested and the Z flag is affected. The destination byte operand is the general-purpose register designated by the Rbd field of the instruction. The bit to be tested is determined from a binary decode of the least significant three bits of a general-purpose word register designated by the Rs field of the instruction. The contents of the destination are unaltered.

Flags

C	Z	S	P/V	DA	H
–	✱	–	–	–	–

Z: Set to 1 if selected bit of destination operand is zero. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
✱ = Conditional – see description

BITB

BIT test in a byte (static)

BITB

BITB dst, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	BITB Rbd, B 1 0 1 0 0 1 1 0 Rbd b	4
IR	NS	BITB Rd↑, B 0 0 1 0 0 1 1 0 Rd ≠ 0 b	8
IR	S	BITB RRd↑, B 0 0 1 0 0 1 1 0 RRd ≠ 0 b	8
DA	NS	BITB LABEL, B 0 1 1 0 0 1 1 0 0 0 0 0 b ADDRESS	10
DA	SSO	BITB LABSSO, B 0 1 1 0 0 1 1 0 0 0 0 0 b 0 SEGMENT OFFSET	11
DA	SLO	BITB LABEL, B 0 1 1 0 0 1 1 0 0 0 0 0 b 1 SEGMENT ▨ OFFSET	13
X	SLO	BITB LABEL (Rx), B 0 1 1 0 0 1 1 0 Rx ≠ 0 b ADDRESS	11
X	NS	BITB LABSSO (Rx), B 0 1 1 0 0 1 1 0 Rx ≠ 0 b 0 SEGMENT OFFSET	11
X	SSO	BITB LABEL (Rx), B 0 1 1 0 0 1 1 0 Rx ≠ 0 b 1 SEGMENT ▨ OFFSET	14

Operation

Z flag←$\overline{\text{byte dst<b bit>}}$

Description

A bit in the byte destination is tested, and the Z flag is affected as shown below. The destination is determined by the applicable addressing mode. The bit to be tested is determined by the binary value of the least significant three bits of the b field of the instruction. The contents of the destination are not altered.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value, b, in the instruction. The range of b is zero through 7, and b = B. Specifying a B outside of the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
–	*	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if specified bit of destination byte is zero. Reset otherwise.

CALL

CALL subroutine

CALL dst

CALL

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CALL Rd↑ 0 0 0 1 1 1 1 1 Rd 0 0 0 0	10
IR	S	CALL RRd↑ 0 0 0 1 1 1 1 1 RRd 0 0 0 0	15
DA	NS	CALL LABEL 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 ADDRESS	12
DA	SSO	CALL LABSSO 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 SEGMENT OFFSET	18
DA	SLO	CALL LABEL 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 SEGMENT ▨ OFFSET	20
X	NS	CALL LABEL (Rx) 0 1 0 1 1 1 1 1 Rx ≠ 0 0 0 0 0 ADDRESS	13
X	SSO	CALL LABSSO (Rx) 0 1 0 1 1 1 1 1 Rx ≠ 0 0 0 0 0 0 SEGMENT OFFSET	18
X	SLO	CALL LABEL (Rx) 0 1 0 1 1 1 1 1 Rx ≠ 0 0 0 0 0 1 SEGMENT ▨ OFFSET	21

Operation (segmented)

R15 <0:15>←R15<0:15>−2
(RR14<0:22>)←Updated PC OFFSET
R15<0:15>←R15<0:15>−2
(RR14)<0:22>)←PC SEGMENT
PC SEGMENT←dst<24:30>
PC OFFSET←dst<0:15>

Operation (non-segmented)

R15<0:15>←R15<0:15>−2
(R15<0:15>)←Updated PC
PC←dst<0:15>

In the system mode the system stack pointer (R15′ or RR14′) is used instead of the normal stack pointer.

Description

The program return address (i.e., the updated contents of PC) is pushed onto the stack addressed by the implied stack pointer register (R15 non-segmented, RR14 segmented). The new program counter address is then loaded to transfer control to the subroutine. The new address is determined by the applicable addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = *Unaffected*
1 = Set
0 = Cleared
✱ = Conditional – see description

CALR

CALL subroutine relative

CALR

CALR LAB

Mode	Version	Mnemonic and Form	Clocks
RA	NS, S	CALR LAB 1 1 0 1 DISPLACEMENT	10, 15

Operation (segmented)

R15 <0:15>←R15<0:15>−2
(RR14<0:22>)←Updated PC OFFSET
R15<0:15>←R15<0:15>−2
(RR14<0:22>)←PC SEGMENT
PC OFFSET←Updated PC OFFSET−
2x displacement

Operation (non-segmented)

R15<0:15>←R15<0:15>−2
(R15<0:15>)←Updated PC
PC←Updated PC−2x displacement

In the system mode the system stack pointer (R15′ or RR14′) is used instead of the normal stack pointer.

Description

The program return address is pushed onto the stack addressed by the implied stack pointer register (R15 non-segmented, RR14 segmented in the normal mode). The signed 12-bit displacement field of the instruction is sign extended and left shifted (word aligned) before being subtracted from the updated PC (return address). The result is then loaded into the program counter to produce a jump address. The program counter segment number remains unaltered. The range of the relative call is −2047 to +2048 words with respect to the updated PC.

Assembler Notation

The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Flags are not affected.

CLEAR word

CLR dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	CLR Rd 1 0 0 0 1 1 0 1 · Rd · 1 0 0 0	7
IR	NS	CLR Rd↑ 0 0 0 0 1 1 0 1 · Rd · 1 0 0 0	8
IR	S	CLR RRd↑ 0 0 0 0 1 1 0 1 · RRd · 1 0 0 0	8
DA	NS	CLR LABEL 0 1 0 0 1 1 0 1 · 0 0 0 0 · 1 0 0 0 ADDRESS	11
DA	SSO	CLR LABSSO 0 1 0 0 1 1 0 1 · 0 0 0 0 · 1 0 0 0 0 SEGMENT · OFFSET	12
DA	SLO	CLR LABEL 0 1 0 0 1 1 0 1 · 0 0 0 0 · 1 0 0 0 1 SEGMENT · ▨ OFFSET	14
X	NS	CLR LABEL (Rx) 0 1 0 0 1 1 0 1 · Rx ≠ 0 · 1 0 0 0 ADDRESS	12
X	SSO	CLR LABSSO (Rx) 0 1 0 0 1 1 0 1 · Rx ≠ 0 · 1 0 0 0 0 SEGMENT · OFFSET	12
X	SLO	CLR LABEL (Rx) 0 1 0 0 1 1 0 1 · Rx ≠ 0 · 1 0 0 0 1 SEGMENT · ▨ OFFSET	15

Operation

dst<0:15> ← 0

Description

The 16 bits of the specified destination word are replaced with zeros. The original contents of the destination are lost. The destination is determined by the applicable addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

CLRB

CLEAR byte

CLRB

CLRB dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	CLRB Rbd 1 0 0 0 1 1 0 0 | Rbd | 1 0 0 0	7
IR	NS	CLRB Rd↑ 0 0 0 0 1 1 0 0 | Rd | 1 0 0 0	8
IR	S	CLRB RRd↑ 0 0 0 0 1 1 0 0 | RRd | 1 0 0 0	8
DA	NS	CLRB LABEL 0 1 0 0 1 1 0 0 | 0 0 0 0 | 1 0 0 0 ADDRESS	11
DA	SSO	CLRB LABSSO 0 1 0 0 1 1 0 0 | 0 0 0 0 | 1 0 0 0 0 | SEGMENT | OFFSET	12
DA	SLO	CLRB LABEL 0 1 0 0 1 1 0 0 | 0 0 0 0 | 1 0 0 0 1 | SEGMENT | ▨ OFFSET	14
X	NS	CLRB LABEL (Rx) 0 1 0 0 1 1 0 0 | Rx ≠ 0 | 1 0 0 0 ADDRESS	12
X	SSO	CLRB LABSSO (Rx) 0 1 0 0 1 1 0 0 | Rx ≠ 0 | 1 0 0 0 0 | SEGMENT | OFFSET	12
X	SLO	CLRB LABEL (Rx) 0 1 0 0 1 1 0 0 | Rx ≠ 0 | 1 0 0 0 1 | SEGMENT | ▨ OFFSET	15

Operation

dst<0:7> ← 0

Description

The eight bits of the specified destination byte are replaced with zeros. The original contents of the destination are lost.The destination is determined by the applicable addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

5

COM

COMPLEMENT word

COM

COM dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	COM Rd 1 0 0 0 1 1 0 1 | Rd | 0 0 0 0	7
IR	NS	COM Rd↑ 0 0 0 0 1 1 0 1 | Rd | 0 0 0 0	12
IR	S	COM RRd↑ 0 0 0 0 1 1 0 1 | RRd | 0 0 0 0	12
DA	NS	COM LABEL 0 1 0 0 1 1 0 1 | 0 0 0 0 | 0 0 0 0 ADDRESS	15
DA	SSO	COM LABSSO 0 1 0 0 1 1 0 1 | 0 0 0 0 | 0 0 0 0 0 | SEGMENT | OFFSET	16
DA	SLO	COM LABEL 0 1 0 0 1 1 0 1 | 0 0 0 0 | 0 0 0 0 1 | SEGMENT | ▨ OFFSET	18
X	NS	COM LABEL (Rx) 0 1 0 0 1 1 0 1 | Rx ≠ 0 | 0 0 0 0 ADDRESS	16
X	SSO	COM LABSSO (Rx) 0 1 0 0 1 1 0 1 | Rx ≠ 0 | 0 0 0 0 0 | SEGMENT | OFFSET	16
X	SLO	COM LABEL (Rx) 0 1 0 0 1 1 0 1 | Rx ≠ 0 | 0 0 0 0 1 | SEGMENT | ▨ OFFSET	19

Operation

$\text{dst}<0:15> \leftarrow \overline{\text{dst}<0:15>}$

Description

The contents of the destination word operand are complemented. The original contents of the destination are lost. The destination operand is determined by the applicable addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	✱	✱	–	–	–

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.

COMB

COMPLEMENT byte

COMB

COMB dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	COMB Rbd 1,0,0,0,1,1,0,0 | Rbd | 0,0,0,0	7
IR	NS	COMB Rd↑ 0,0,0,0,1,1,0,0 | Rd | 0,0,0,0	12
IR	S	COMB RRd↑ 0,0,0,0,1,1,0,0 | RRd | 0,0,0 0	12
DA	NS	COMB LABEL 0,1,0,0,1,1,0,0 | 0,0,0,0 | 0,0,0,0 ADDRESS	15
DA	SSO	COMB LABSSO 0,1,0,0,1,1,0,0 | 0,0,0,0 | 0,0,0,0 0 | SEGMENT | OFFSET	16
DA	SLO	COMB LABEL 0,1,0,0,1,1,0,0 | 0,0,0,0 | 0,0,0,0 1 | SEGMENT | ▨ OFFSET	18
X	NS	COMB LABEL (Rx) 0,1,0,0,1,1,0,0 | Rx ≠ 0 | 0,0,0,0 ADDRESS	16
X	SSO	COMB LABSSO (Rx) 0,1,0,0,1,1,0,0 | Rx ≠ 0 | 0,0,0,0 0 | SEGMENT | OFFSET	16
X	SLO	COMB LABEL (Rx) 0,1,0,0,1,1,0,0 | Rx ≠ 0 | 0,0,0,0 1 | SEGMENT | ▨ OFFSET	19

Operation

$dst<0:7> \leftarrow \overline{dst<0:7>}$

Description

The contents of the destination byte operand are complemented. The original contents of the destination are lost. The destination operand is determined by the applicable addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	✱	✱	✱	–	–

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if parity of result is even. Reset otherwise.

COMFLG

COMPLEMENT FLAGS

COMFLG

COMFLG LIST

Mode	Version	Mnemonic and Form	Clocks	Operation
—	NS, S	COMFLG LIST 1 0 0 0 1 1 0 1 C Z S PV 0 1 0 1	7	$FCW<C;Z;S;P/V> \leftarrow \overline{FCW<C;Z;S;P/V>}$ (See description below)

Description

The CPU flags C, Z, S and P/V are complemented or unaltered, according to the bit settings in the instruction field as described in the table below.

Instruction Bit	If = 0	If = 1	Assembler Notation
7	No Effect	Complement C Flag	CY
6	No Effect	Complement Z Flag	ZR
5	No Effect	Complement S Flag	SGN
4	No Effect	Complement P/V Flag	PY or OV

Assembler Notation

The assembler notation LIST refers to a list of any or all of the following reserved words, separated by commas: CY, ZR, SGN, PY or OV. Note that PY and OV affect the same flag (P/V).

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	*

See above.
H: Undefined.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

CP

COMPARE register with word

CP

CP Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	CP Rd, Rs 1 0 0 0 1 0 1 1 Rs Rd	4
IM	NS, S	CP Rd, IM 0 0 0 0 1 0 1 1 0 0 0 0 Rd OPERAND	7
IR	NS	CP Rd, Rs↑ 0 0 0 0 1 0 1 1 Rs ≠ 0 Rd	7
IR	S	CP Rd, RRs↑ 0 0 0 0 1 0 1 1 RRs ≠ 0 Rd	7
DA	NS	CP Rd, LABEL 0 1 0 0 1 0 1 1 0 0 0 0 Rd ADDRESS	9
DA	SSO	CP Rd, LABSSO 0 1 0 0 1 0 1 1 0 0 0 0 Rd 0 SEGMENT OFFSET	10
DA	SLO	CP Rd, LABEL 0 1 0 0 1 0 1 1 0 0 0 0 Rd 1 SEGMENT ▨ OFFSET	12
X	NS	CP Rd, LABEL (Rx) 0 1 0 0 1 0 1 1 Rx ≠ 0 Rd ADDRESS	10
X	SSO	CP Rd, LABSSO (Rx) 0 1 0 0 1 0 1 1 Rx ≠ 0 Rd 0 SEGMENT OFFSET	10
X	SLO	CP Rd, LABEL (Rx) 0 1 0 0 1 0 1 1 Rx ≠ 0 Rd 1 SEGMENT ▨ OFFSET	13

Operation

Use the result of Rd<0:15>−src<0:15> to set flags.

Description

The source word operand is compared by subtraction with the contents of a general-purpose word register designated by the Rd field of the instruction. The source operand is determined by the applicable addressing mode. Both the source contents and destination contents are unaltered.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = *Unaffected*
1 = Set
0 = Cleared
* = Conditional − see description

C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CP COMPARE IMMEDIATE word with memory CP

CP dst, IM

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CP Rd↑, IM 0 0 0 0 1 1 0 1 │ Rd │ 0 0 0 1 OPERAND	11
IR	S	CP RRd↑, IM 0 0 0 0 1 1 0 1 │ RRd │ 0 0 0 1	14
DA	NS	CP LABEL, IM 0 1 0 0 1 1 0 1 │ 0 0 0 0 │ 0 0 0 1 ADDRESS OPERAND	14
DA	SSO	CP LABSSO, IM 0 1 0 0 1 1 0 1 │ 0 0 0 0 │ 0 0 0 1 0 │ SEGMENT │ OFFSET OPERAND	15
DA	SLO	CP LABEL, IM 0 1 0 0 1 1 0 1 │ 0 0 0 0 │ 0 0 0 1 1 │ SEGMENT │ ▨ OFFSET OPERAND	17
X	NS	CP LABEL (Rx), IM 0 1 0 0 1 1 0 1 │ Rx ≠ 0 │ 0 0 0 1 ADDRESS OPERAND	15
X	SSO	CP LABSSO (Rx), IM 0 1 0 0 1 1 0 1 │ Rx ≠ 0 │ 0 0 0 1 0 │ SEGMENT │ OFFSET OPERAND	15
X	SLO	CP LABEL (Rx), IM 0 1 0 0 1 1 0 1 │ Rx ≠ 0 │ 0 0 0 1 1 │ SEGMENT │ ▨ OFFSET OPERAND	18

Operation

Use result of dst<0:15> − src<0:15> to set flags (see below).

Description

The immediate source word operand is compared with the destination word operand. The comparison is achieved by subtraction. The destination operand is determined by the applicable addressing mode. The contents of the destination operand are unaltered and the only action is to set the flags as described below.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

COMPARE register with byte

CPB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	CPB Rbd, Rbs 1 0 0 0 1 0 1 0 │ Rbs │ Rbd	4
IM	NS, S	CPB Rbd, IMb 0 0 0 0 1 0 1 0 │ 0 0 0 0 │ Rbd 7 OPERAND 0 │ 7 OPERAND 0	7
IR	NS	CPB Rbd, Rs↑ 0 0 0 0 1 0 1 0 │ Rs ≠ 0 │ Rbd	7
IR	S	CPB Rbd, RRs↑ 0 0 0 0 1 0 1 0 │ RRs ≠ 0 │ Rbd	7
DA	NS	CPB Rbd, LABEL 0 1 0 0 1 0 1 0 │ 0 0 0 0 │ Rbd ADDRESS	9
DA	SSO	CPB Rbd, LABSSO 0 1 0 0 1 0 1 0 │ 0 0 0 0 │ Rbd 0 │ SEGMENT │ OFFSET	10
DA	SLO	CPB Rbd, LABEL 0 1 0 0 1 0 1 0 │ 0 0 0 0 │ Rbd 1 │ SEGMENT │ ▨ OFFSET	12
X	NS	CPB Rbd, LABEL (Rx) 0 1 0 0 1 0 1 0 │ Rx ≠ 0 │ Rbd ADDRESS	10
X	SSO	CPB Rbd, LABSSO (Rx) 0 1 0 0 1 0 1 0 │ Rx ≠ 0 │ Rbd 0 │ SEGMENT │ OFFSET	10
X	SLO	CPB Rbd, LABEL (Rx) 0 1 0 0 1 0 1 0 │ Rx ≠ 0 │ Rbd 1 │ SEGMENT │ ▨ OFFSET	13

Operation

Use result of Rbd<0:7>−src<0:7> to set flags.

Description

The source byte operand is compared by subtraction with the contents of a general-purpose byte register designated by the Rbd field of the instruction. The source operand is determined by the applicable addressing mode. Both the source contents and destination contents are unaltered.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CPB — COMPARE IMMEDIATE byte with memory — CPB

CPB dst, IMb

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CPB Rd↑, IMb 0,0,0,0,1,1,0,0 | Rd | 0,0,0,1 7 OPERAND 0 | 7 OPERAND 0	11
IR	S	CPB RRd↑, IMb 0,0,0,0,1,1,0,0 | RRd | 0,0,0,1	11
DA	NS	CPB LABEL, IMb 0,1,0,0,1,1,0,0 | 0,0,0,0 | 0,0,0,1 ADDRESS 7 OPERAND 0 | 7 OPERAND 0	14
DA	SSO	CPB LABSSO, IMb 0,1,0,0,1,1,0,0 | 0,0,0,0 | 0,0,0,1 0 | SEGMENT | OFFSET 7 OPERAND 0 | 7 OPERAND 0	15
DA	SLO	CPB LABEL, IMb 0,1,0,0,1,1,0,0 | 0,0,0,0 | 0,0,0,1 1 | SEGMENT | ▨ OFFSET 7 OPERAND 0 | 7 OPERAND 0	17
X	NS	CPB LABEL (Rx), IMb 0,1,0,0,1,1,0,0 | Rx ≠ 0 | 0,0,0,1 ADDRESS 7 OPERAND 0 | 7 OPERAND 0	15
X	SSO	CPB LABSSO (Rx), IMb 0,1,0,0,1,1,0,0 | Rx ≠ 0 | 0,0,0,1 0 | SEGMENT | OFFSET 7 OPERAND 0 | 7 OPERAND 0	15
X	SLO	CPB LABEL (Rx), IMb 0,1,0,0,1,1,0,0 | Rx ≠ 0 | 0,0,0,1 1 | SEGMENT | ▨ OFFSET 7 OPERAND 0 | 7 OPERAND 0	18

Operation

Use result of dst<0:7> − src<0:7> to set flags.

Description

The immediate source byte operand is compared by subtraction with the destination byte operand. The destination operand is determined by the applicable addressing mode. The contents of the destination operand are unaltered.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CPD

COMPARE register to memory word, autodecrement

CPD

CPD Rd, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPD Rd, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 1		Rs	1 0 0 0	20
		0 0 0 0	Rc	Rd	CC	
		CPD Rd, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 1		RRs	1 0 0 0	20
		0 0 0 0	Rc	Rd	CC	

Operation

If result of Rd<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−2
Rc<0:15>←Rc<0:15>−1

Description

The source word operand is compared to the destination word operand by subtraction. The destination operand is the contents of the general-purpose word register designated by the Rd field of the instruction. The source operand is a word in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. Both source and destination operands are unaltered, and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by two.

R0 can be designated as the general-purpose source register.

5

Flags

C	Z	S	P/V	DA	H
−	∗	−	∗	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

COMPARE register to memory byte, autodecrement

CPDB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CPDB Rbd, Rs↑, Rc, CC	20
IR	S	CPDB Rbd, RRs↑, Rc, CC	20

CPDB Rbd, Rs↑, Rc, CC

1 0 1 1	1 0 1 0	Rs	1 0 0 0
0 0 0 0	Rc	Rbd	CC

CPDB Rbd, RRs↑, Rc, CC

1 0 1 1	1 0 1 0	RRs	1 0 0 0
0 0 0 0	Rc	Rbd	CC

Operation

If result of Rbd<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−1
Rc<0:15>←Rc<0:15>−1

Description

The source byte operand is compared to the destination byte operand by subtraction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. The source operand is a byte in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. Both source and destination operands are unaltered, and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by one.

R0 can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

CPDR

COMPARE register to memory word, autodecrement and repeat

CPDR

CPDR Rd, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CPDR Rd, Rs↑, Rc, CC 1 0 1 1 1 0 1 1 │ Rs │ 1 1 0 0 0 0 0 0 │ Rc │ Rd │ CC	11 + 9n*
IR	S	CPDR Rd, RRs↑, Rc, CC 1 0 1 1 1 0 1 1 │ RRs │ 1 1 0 0 0 0 0 0 │ Rc │ Rd │ CC	11 + 9n*

*n is the number of iterations.

Operation

If Rd<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source word operand is compared to the destination word operand by subtraction. The source operand is a word in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the contents of the general-purpose word register designated by the Rd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by two, and the operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
–	∗	–	∗	–	–

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional − see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

CPDRB

COMPARE register to memory byte, autodecrement and repeat

CPDRB

CPDRB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPDRB Rbd, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 0		Rs	1 1 0 0	11 + 9n*
		0 0 0 0	Rc	Rbd	CC	
		CPDRB Rbd, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 0		RRs	1 1 0 0	11 + 9n*
		0 0 0 0	Rc	Rbd	CC	

*n is the number of iterations.

Operation

If Rbd<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source byte operand is compared to the destination byte operand by subtraction. The source operand is a byte in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are decremented by one, and the operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

COMPARE register to memory word, autoincrement

CPI Rd, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CPI Rd, Rs↑, Rc, CC	20

1 0 1 1	1 0 1 1	Rs	0 0 0 0
0 0 0 0	Rc	Rd	CC

Mode	Version	Mnemonic and Form	Clocks
IR	S	CPI Rd, RRs↑, Rc, CC	20

1 0 1 1	1 0 1 1	RRs	0 0 0 0
0 0 0 0	Rc	Rd	CC

Operation

If result of Rd<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+2
Rc<0:15>←Rc<0:15>−1

Description

The source word operand is compared to the destination word operand by subtraction. The destination operand is the contents of the general-purpose word register designated by the Rd field of the instruction. The source operand is a word in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. Both the source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by two.

R0 can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
−	∗	−	∗	−	−

− = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional − see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

CPIB

COMPARE register to memory byte, autoincrement

CPIB

CPIB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPIB Rbd, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 0	Rs	0 0 0 0		20
		0 0 0 0	Rc	Rbd	CC	
		CPIB Rbd, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 0	RRs	0 0 0 0		20
		0 0 0 0	Rc	Rbd	CC	

Operation

If result of Rbd<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

The source byte operand is compared to the destination byte operand by subtraction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. The source operand is a byte in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. Both the source and destination operands are unaltered, and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by one.

R0 can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

CPIR

COMPARE register to memory word, autoincrement and repeat

CPIR

CPIR Rd, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks
		CPIR Rd, Rs↑, Rc, CC	
IR	NS	1 0 1 1 1 0 1 1 | Rs | 0 1 0 0 0 0 0 0 | Rc | Rd | CC	11 + 9n*
		CPIR Rd, RRs↑, Rc, CC	
IR	S	1 0 1 1 1 0 1 1 | RRs | 0 1 0 0 0 0 0 0 | Rc | Rd | CC	11 + 9n*

*n is the number of iterations.

Operation

If Rd<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source word operand is compared to the destination word operand by subtraction. The source operand is a word in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the content of the general-purpose word register designated by the Rd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by two. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

5

Flags

C	Z	S	P/V	DA	H
–	∗	–	∗	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

CPIRB

COMPARE register to memory byte, autoincrement and repeat

CPIRB

CPIRB Rbd, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPIRB Rbd, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 0		Rs	0 1 0 0	11 + 9n*
		0 0 0 0	Rc	Rbd	CC	
		CPIRB Rbd, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 0		RRs	0 1 0 0	11 + 9n*
		0 0 0 0	Rc	Rbd	CC	

*n is the number of iterations.

Operation

If Rbd<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source byte operand is compared to the destination byte operand by subtraction. The source operand is a byte in memory addressed by the general-purpose register designated by the Rs (or RRs) field of the instruction. The destination operand is the contents of the general-purpose byte register designated by the Rbd field of the instruction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs are incremented by one, and the operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

CPL

COMPARE register with long word

CPL

CPL RRd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	CPL RRd, RRs 1 0 0 1 0 0 0 0 │ RRs │ RRd	8
IM	NS, S	CPL RRd, IMℓ 0 0 0 1 0 0 0 0 │ 0 0 0 0 │ RRd 31 OPERAND 16 15 OPERAND 0	14
IR	NS	CPL RRd, Rs↑ 0 0 0 1 0 0 0 0 │ Rs ≠ 0 │ RRd	14
IR	S	CPL RRd, RRs↑ 0 0 0 1 0 0 0 0 │ RRs ≠ 0 │ RRd	14
DA	NS	CPL RRd, LABEL 0 1 0 1 0 0 0 0 │ 0 0 0 0 │ RRd ADDRESS	15
DA	SSO	CPL RRd, LABSSO 0 1 0 1 0 0 0 0 │ 0 0 0 0 │ RRd 0 │ SEGMENT │ OFFSET	16
DA	SLO	CPL RRd, LABEL 0 1 0 1 0 0 0 0 │ 0 0 0 0 │ RRd 1 │ SEGMENT │ ▨ OFFSET	18
X	NS	CPL RRd, LABEL (Rx) 0 1 0 1 0 0 0 0 │ Rx ≠ 0 │ RRd ADDRESS	16
X	SSO	CPL RRd, LABSSO (Rx) 0 1 0 1 0 0 0 0 │ Rx ≠ 0 │ RRd 0 │ SEGMENT │ OFFSET	16
X	SLO	CPL RRd, LABEL (Rx) 0 1 0 1 0 0 0 0 │ Rx ≠ 0 │ RRd 1 │ SEGMENT │ ▨ OFFSET	19

Operation

Use result of RRd<0:31> − src<0:31> to set flags.

Description

The source long word operand is compared by subtraction with the contents of a general-purpose register pair designated by the RRd field of the instruction. The source operand is determined by the applicable addressing mode. Both the source contents and destination contents are unaltered.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

C: Reset on carry from most significant bit of result. Otherwise set to 1, indicating a borrow.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

CPSD

COMPARE word strings in memory, autodecrement

CPSD

CPSD dst, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPSD Rd↑, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 1		Rs	1 0 1 0	25
		0 0 0 0	Rc	Rd	CC	
		CPSD RRd↑, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 1		RRs	1 0 1 0	25
		0 0 0 0	Rc	RRd	CC	

Operation

If result of dst<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−2
Rd<0:15>←Rd<0:15>−2
Rc<0:15>←Rc<0:15>−1

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rd and Rs (or RRd and RRs) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The source and destination operands are unaltered. The contents of the Rs and Rd registers are decremented by two.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

CPSDB COMPARE byte strings in memory, autodecrement CPSDB

CPSDB dst, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
IR	NS	CPSDB Rd↑, Rs↑, Rc, CC				25
		1,0,1,1	1,0,1,0	Rs	1,0,1,0	
		0,0,0,0	Rc	Rd	CC	
IR	S	CPSDB RRd↑, RRs↑, Rc, CC				25
		1,0,1,1	1,0,1,0	RRs	1,0,1,0	
		0,0,0,0	Rc	RRd	CC	

Operation

If result of dst<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−1
Rd<0:15>←Rd<0:15>−1
Rc<0:15>←Rc<0:15>−1

Description

The source byte operand is compared to the destination byte operand. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rd and Rs (or RRd and RRs) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by one. The source and destination operands are unaltered.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

CPSDR **CPSDR**

COMPARE word strings in memory, autodecrement and repeat

CPSDR dst, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPSDR Rd↑, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 1		Rs	1 1 1 0	11 + 14n*
		0 0 0 0	Rc	Rd	CC	
		CPSDR RRd↑, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 1		RRs	1 1 1 0	11 + 14n*
		0 0 0 0	Rc	RRd	CC	

*n is the number of iterations.

Operation

If result of dst<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−2
Rd<0:15>←Rd<0:15>−2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rd and Rs (or RRd and RRs) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both decremented by two. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	✱	–	✱	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

CPSDRB — COMPARE byte strings in memory, autodecrement and repeat — CPSDRB

CPSDRB dst, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CPSDRB Rd↑, Rs↑, Rc, CC	11 + 14n*
IR	S	CPSDRB RRd↑, RRs↑, Rc, CC	11 + 14n*

CPSDRB Rd↑, Rs↑, Rc, CC

1 0 1 1	1 0 1 0	Rs	1 1 1 0
0 0 0 0	Rc	Rd	CC

CPSDRB RRd↑, RRs↑, Rc, CC

1 0 1 1	1 0 1 0	RRs	1 1 1 0
0 0 0 0	Rc	RRd	CC

*n is the number of iterations.

Operation

If result of dst<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>−1
Rd<0:15>←Rd<0:15>−1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source byte operand is compared to the destination byte operand. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both decremented by one. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	∗	–	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

CPSI

COMPARE word strings in memory, autoincrement

CPSI

CPSI dst, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPSI Rd↑, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 1		Rs	0 0 1 0	25
		0 0 0 0	Rc	Rd	CC	
		CPSI RRd↑, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 1		RRs	0 0 1 0	25
		0 0 0 0	Rc	RRd	CC	

Operation

If result of dst<0:15> − src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+2
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The source and destination operands are unaltered. The contents of the Rs and Rd registers are incremented by two.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	✱	–	✱	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
✱ = Conditional – see description

CPSIB **CPSIB**

COMPARE byte strings in memory, autoincrement

CPSIB dst, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPSIB Rd↑, Rs↑, Rc, CC				
IR	NS	1 0 1 1	1 0 1 0	Rs	0 0 1 0	25
		0 0 0 0	Rc	Rd	CC	
		CPSIB RRd↑, RRs↑, Rc, CC				
IR	S	1 0 1 1	1 0 1 0	RRs	0 0 1 0	25
		0 0 0 0	Rc	RRd	CC	

Operation

If dst<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+1
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

The source byte operand is compared to the destination byte operand by subtraction. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. Both source and destination operands are unaltered. The contents of the Rs and Rd registers are incremented by one.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	∗	–	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

CPSIR

COMPARE word strings in memory, autoincrement and repeat

CPSIR

CPSIR dst, src, Rc, CC

Mode	Version	Mnemonic and Form				Clocks
		CPSIR Rd↑, Rs↑, Rc, CC				
IR	NS	1 0 1 1 1 0 1 1		Rs	0 1 1 0	11 + 14n*
		0 0 0 0	Rc	Rd	CC	
		CPSIR RRd↑, RRs↑, Rc, CC				
IR	S	1 0 1 1 1 0 1 1		RRs	0 1 1 0	11 + 14n*
		0 0 0 0	Rc	RRd	CC	

*n is the number of iterations.

Operation

If result of dst<0:15>−src<0:15> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+2
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source word operand is compared to the destination word operand. Both the source and destination operands are words in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both incremented by two. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

CPSIRB

COMPARE byte strings in memory, autoincrement and repeat

CPSIRB

CPSIRB dst, src, Rc, CC

Mode	Version	Mnemonic and Form	Clocks
IR	NS	CPSIRB Rd↑, Rs↑, Rc, CC	11 + 14n*
IR	S	CPSIRB RRd↑, RRs↑, Rc, CC	11 + 14n*

CPSIRB Rd↑, Rs↑, Rc, CC

1 0 1 1	1 0 1 0	Rs	0 1 1 0
0 0 0 0	Rc	Rd	CC

CPSIRB RRd↑, RRs↑, Rc, CC

1 0 1 1	1 0 1 0	RRs	0 1 1 0
0 0 0 0	Rc	RRd	CC

*n is the number of iterations.

Operation

If dst<0:7>−src<0:7> meets CC condition in instruction, then Z flag←1.
Rs<0:15>←Rs<0:15>+1
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source byte operand is compared to the destination byte operand. Both the source and destination operands are bytes in memory addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The comparison is achieved by subtraction. Both source and destination operands are unaltered and the only action is to set the flags. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of the Rs and Rd registers are both incremented by one. The operation will repeat until termination.

Termination occurs when either the contents of Rc are zero or CC condition is met. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

5

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if a comparison matches condition specified in CC field. Reset otherwise.
P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

DAB

DECIMAL ADJUST byte

DAB

DAB Rbd

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	DAB Rbd 1 0 1 1 0 0 0 0 Rbd 0 0 0 0	5	Rbd<0:7>←Rbd<0:7>+BCD<0:7>

Description

A destination byte register, designated by the Rd field of the instruction, is adjusted by the addition of the BCD operand given in the table below. This instruction converts a byte (binary representation) into a two digit binary coded decimal representation, following an arithmetic operation.

Preceding Arithmetic Operation	C Flag Before DAB	dst<4:7> (Hex)	H Flag Before DAB	dst<0:3> (Hex)	BCD<0:7>	C Flag After DAB
	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
ADDB	0	0-9	1	0-3	06	0
ADCB	0	A-F	0	0-9	60	1
	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
	0	0-9	0	0-9	00	0
SUBB	0	0-8	1	6-F	FA	0
SBCB	1	7-F	0	0-9	A0	1
	1	6-F	1	6-F	9A	1

Flags

C	Z	S	P/V	DA	H
*	*	*	–	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set or reset according to table.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if the most significant bit of the result is set. Reset otherwise.

DECREMENT byte register and jump on non-zero

DBJNZ Rbc, LAB

Mode	Version	Mnemonic and Form	Clocks
		DBJNZ Rbc, LAB	
RA	NS, S	1 1 1 1 | Rbc | 0 | DISPLACEMENT	11

Operation

Rbc<0:7>←Rbc<0:7>−1
If Rc<0:7>≠0,
then PC←Updated Pc−2x displacement.
Otherwise PC←Updated PC.

Description

The contents of the general-purpose byte register designated by the Rbc field of the instruction are decremented, and if this produces a non-zero result, a jump is executed. The jump address is obtained by subtracting the contents of the 7-bit displacement field, which has been left shifted (i.e., word aligned) from the contents of the updated program counter (i.e., incremented by two). The resultant address is loaded into the program counter and is used as the jump destination. The instruction displacement field is interpreted as a 7-bit unsigned integer. Thus the range of the relative jump is zero to −127 words with respect to the updated PC.

If the register decrementation produces a zero result, then the contents of the program counter are merely updated by incrementing by two.

Assembler Notation

The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

DEC

DECREMENT word

DEC

DEC dst, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	DEC Rd, N 1 0 1 0 1 0 1 1 │ Rd │ n	4
IR	NS	DEC Rd↑, N 0 0 1 0 1 0 1 1 │ Rd │ n	11
IR	S	DEC RRd↑, N 0 0 1 0 1 0 1 1 │ RRd │ n	11
DA	NS	DEC LABEL, N 0 1 1 0 1 0 1 1 │ 0 0 0 0 │ n ADDRESS	13
DA	SSO	DEC LABSSO, N 0 1 1 0 1 0 1 1 │ 0 0 0 0 │ n 0 │ SEGMENT │ OFFSET	14
DA	SLO	DEC LABEL, N 0 1 1 0 1 0 1 1 │ 0 0 0 0 │ n 1 │ SEGMENT │ ▨ OFFSET	16
X	NS	DEC LABEL (Rx), N 0 1 1 0 1 0 1 1 │ Rx ≠ 0 │ n ADDRESS	14
X	SSO	DEC LABSSO (Rx), N 0 1 1 0 1 0 1 1 │ Rx ≠ 0 │ n 0 │ SEGMENT │ OFFSET	14
X	SLO	DEC LABEL (Rx), N 0 1 1 0 1 0 1 1 │ Rx ≠ 0 │ n 1 │ SEGMENT │ ▨ OFFSET	17

Operation

dst<0:15>←dst<0:15>−N

Description

A value between 1 and 16 is subtracted from the destination operand word and the result is loaded back into the destination. The desired value to be subtracted is specified by the n field, where n = 0 corresponds to value one and so on, and n = F corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is 1 to 16, and n = N − 1. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

DECB

DECREMENT byte

DECB

DECB dst, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	DECB Rbd, N 1 0 1 0 1 0 1 0 Rbd n	4
IR	NS	DECB Rd↑, N 0 0 1 0 1 0 1 0 Rd n	11
IR	S	DECB RRd↑, N 0 0 1 0 1 0 1 0 RRd n	11
DA	NS	DECB LABEL, N 0 1 1 0 1 0 1 0 0 0 0 0 n ADDRESS	13
DA	SSO	DECB LABSSO, N 0 1 1 0 1 0 1 0 0 0 0 0 n 0 SEGMENT OFFSET	14
DA	SLO	DECB LABEL, N 0 1 1 0 1 0 1 0 0 0 0 0 n 1 SEGMENT ▨ OFFSET	16
X	NS	DECB LABEL (Rx), N 0 1 1 0 1 0 1 0 Rx ≠ 0 n ADDRESS	14
X	SSO	DECB LABSSO (Rx), N 0 1 1 0 1 0 1 0 Rx ≠ 0 n 0 SEGMENT OFFSET	14
X	SLO	DECB LABEL (Rx), N 0 1 1 0 1 0 1 0 Rx ≠ 0 n 1 SEGMENT ▨ OFFSET	17

Operation

dst<0:7>←dst<0:7>−N

Description

A value between 1 and 16 is subtracted from the destination byte operand and the result is loaded back into the destination. The desired value to be subtracted is specified by the n field, where n = 0 corresponds to value one and so on, and n = F corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is 1 to 16, and n = N − 1. Specifying an N outside of the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
−	∗	∗	∗	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow.

DI

DISABLE INTERRUPT

DI

DI LIST

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
		DI LIST	
—	NS, S	0 1 1 1 1 1 0 0 0 0 0 0 0 0 V N	7

Operation

FCW<11>←0 for N=0
FCW<11>←FCW<11> for N=1
FCW<12>←0 for V=0
FCW<12>←FCW<12> for V=1

Description

The interrupt enables in the FCW are reset to zero depending upon the values of the N and V bits within the instruction. A value of one in these bit positions causes the relevant interrupt enable to be unaltered, and a value of zero causes the relevant interrupt enable to be Reset. The bit designated V in the instruction controls the vectored interrupt enable bit and the bit designated N controls the non-vectored interrupt enable bit.

Instruction Bit	FCW Bit/#	Assembler Notation
0	NVIE/11	NVI
1	VIE/12	VI

Assembler Notation

The assembler notation LIST refers to a list of either or both of the following reserved words, separated by commas: NVI, VI. Specifying an entry disables that interrupt (i.e., resets the FCW bit).

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

DIV

DIVIDE register pair by source word

DIV

DIV RRd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	DIV RRd, Rs 1 0 0 1 1 0 1 1 | Rs | RRd	107
IM	NS, S	DIV RRd, IM 0 0 0 1 1 0 1 1 | 0 0 0 0 | RRd OPERAND	107
IR	NS	DIV RRd, Rs↑ 0 0 0 1 1 0 1 1 | Rs ≠ 0 | RRd	107
IR	S	DIV RRd, RRs↑ 0 0 0 1 1 0 1 1 | RRs ≠ 0 | RRd	107
DA	NS	DIV RRd, LABEL 0 1 0 1 1 0 1 1 | 0 0 0 0 | RRd ADDRESS	108
DA	SSO	DIV RRd, LABSSO 0 1 0 1 1 0 1 1 | 0 0 0 0 | RRd 0 | SEGMENT | OFFSET	109
DA	SLO	DIV RRd, LABEL 0 1 0 1 1 0 1 1 | 0 0 0 0 | RRd 1 | SEGMENT | ▨ OFFSET	111
X	NS	DIV RRd, LABEL (Rx) 0 1 0 1 1 0 1 1 | Rx ≠ 0 | RRd ADDRESS	109
X	SSO	DIV RRd, LABSSO (Rx) 0 1 0 1 1 0 1 1 | Rx ≠ 0 | RRd 0 | SEGMENT | OFFSET	109
X	SLO	DIV RRd, LABEL (Rx) 0 1 0 1 1 0 1 1 | Rx ≠ 0 | RRd 1 | SEGMENT | ▨ OFFSET	112

Operation

RRd<0:15>←RRd<0:31>/src<0:15>
RRd<16:31>←Remainder

Description

A 32-bit signed integer (dividend) is contained in a destination register pair designated by the RRd field of the instruction.

A 16-bit signed integer source operand (divisor) is determined by the applicable addressing mode. Division is performed to obtain a 16-bit quotient and a 16-bit remainder.

The quotient is loaded into the least significant destination register. The remainder is loaded into the most significant destination register. The source operand is not altered.

The original contents of the destination are lost unless the division operation is aborted. This occurs if the divisor is zero or if the magnitude of the divisor is less than or equal to the magnitude of the high order half of the dividend.

The aborted instruction takes less than 30 clock cycles.

5

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if the quotient is less than -2^{15} or greater than/equal to 2^{15} Reset otherwise.
Z: Set to 1 if either the quotient or divisor is zero. Reset otherwise.
S: Set to 1 if quotient is negative. Reset otherwise.
P/V: Set to 1 if division is aborted. Reset otherwise.

DIVL

DIVIDE register quadruple by source long word

DIVL

DIVL RQd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	DIVL RQd, RRs 1 0 0 1 1 0 1 0 | RRs | RQd	744
IM	NS, S	DIVL RQd, IMℓ 0 0 0 1 1 0 1 0 | 0 0 0 0 | RQd 31 OPERAND 16 15 OPERAND 0	744
IR	NS	DIVL RQd, Rs↑ 0 0 0 1 1 0 1 0 | Rs ≠ 0 | RQd	744
IR	S	DIVL RQd, RRs↑ 0 0 0 1 1 0 1 0 | RRs ≠ 0 | RQd	744
DA	NS	DIVL RQd, LABEL 0 1 0 1 1 0 1 0 | 0 0 0 0 | RQd ADDRESS	745
DA	SSO	DIVL RQd, LABSSO 0 1 0 1 1 0 1 0 | 0 0 0 0 | RQd 0 | SEGMENT | OFFSET	746
DA	SLO	DIVL RQd, LABEL 0 1 0 1 1 0 1 0 | 0 0 0 0 | RQd 1 | SEGMENT | ▨ OFFSET	748
X	NS	DIVL RQd, LABEL (Rx) 0 1 0 1 1 0 1 0 | Rx ≠ 0 | RQd ADDRESS	746
X	SSO	DIVL RQd, LABSSO (Rx) 0 1 0 1 1 0 1 0 | Rx ≠ 0 | RQd 0 | SEGMENT | OFFSET	746
X	SLO	DIVL RQd, LABEL (Rx) 0 1 0 1 1 0 1 0 | Rx ≠ 0 | RQd 1 | SEGMENT | ▨	749

Operation

RQd<0:31>←RQd<0:63>/src<0:31>
RQd<32:63>←Remainder

Description

A 64-bit signed integer (dividend) is contained in a quadruple destination register designated by the RQd field of the instruction.

A 32-bit signed integer source operand (divisor) is determined by the applicable addressing mode. Division is performed to obtain a 32-bit quotient and a 32-bit remainder.

The quotient is loaded into the least significant destination register pair. The remainder is loaded into the most significant destination register pair. The source operand is not altered.

The original contents of the destination are lost unless the division operation is aborted. This occurs if the divisor is zero or if the magnitude of the divisor is less than or equal to the magnitude of the high order half of the dividend.

The aborted instruction takes a maximum of 60 clock cycles.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if the quotient is less than -2^{31} or greater than/equal to 2^{31}. Reset otherwise.
Z: Set to 1 if either the quotient or divisor is zero. Reset otherwise.
S: Set to 1 if quotient is negative. Reset otherwise.
P/V: Set to 1 if division is aborted. Reset otherwise.

DJNZ

DECREMENT word register and jump on non-zero

DJNZ

DJNZ Rc, LAB

Mode	Version	Mnemonic and Form	Clocks
RA	NS, S	DJNZ Rc, LAB 1 1 1 1 | Rc | 1 | DISPLACEMENT	11

Operation

Rc<0:15>←Rc<0:15>−1
If Rc<0:15>≠0,
then PC←Updated Pc−2x displacement.
Otherwise PC←Updated PC.

Description

The contents of the general-purpose word register designated by the Rc field of the instruction are decremented and if this produces a non-zero result, a jump is executed. The jump address is obtained by subtracting the contents of the 7-bit instruction displacement field which has been left shifted (i.e., word aligned) from the contents of the updated program counter (i.e., incremented by two). The resultant address is loaded into the program counter and is used as the jump destination. The displacement field is interpreted as a 7-bit unsigned integer. Thus the range of the relative jump is zero to −127 words with respect to the updated PC.

If the register decrementation produces a zero result, then the contents of the program counter are merely updated by incrementing by two.

Assembler Notation

The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

ENABLE INTERRUPT

EI LIST

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
		EI LIST	
—	NS, S	0 1 1 1 1 1 0 0 | 0 0 0 0 0 1 V N	7

Operation

FCW<11>←1 for N=0
FCW<11>←FCW<11> for N=1
FCW<12>←1 for V=0
FCW<12>←FCW<12> for V=1

Description

The interrupt enables in the FCW are set to one depending upon the values of the N and V bits within the instruction. A value of one in these bit positions causes the relevant interrupt enable to be unaltered, and a value of zero causes the relevant interrupt enable to be set. The bit designated V in the instruction controls the vectored interrupt enable bit and the bit designated N controls the non-vectored interrupt enable bit.

Instruction Bit	FCW Bit/#	Assembler Notation
0	NVIE/11	NVI
1	VIE/12	VI

Assembler Notation

The assembler notation LIST refers to a list of either or both of the following reserved words, separated by commas: NVI, VI. Specifying an entry enables that interrupt (i.e., sets the FCW bit).

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

EX

EXCHANGE source word with destination word

EX

EX Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	EX Rd, Rs 1 0 1 0 1 1 0 1 │ Rs │ Rd	6
IR	NS	EX Rd, Rs↑ 0 0 1 0 1 1 0 1 │ Rs │ Rd	12
IR	S	EX Rd, RRs↑ 0 0 1 0 1 1 0 1 │ RRs │ Rd	12
DA	NS	EX Rd, LABEL 0 1 1 0 1 1 0 1 │ 0 0 0 0 │ Rd ADDRESS	15
DA	SSO	EX Rd, LABSSO 0 1 1 0 1 1 0 1 │ 0 0 0 0 │ Rd 0 │ SEGMENT │ OFFSET	16
DA	SLO	EX Rd, LABEL 0 1 1 0 1 1 0 1 │ 0 0 0 0 │ Rd 1 │ SEGMENT │ ▨ OFFSET	18
X	NS	EX Rd, LABEL (Rx) 0 1 1 0 1 1 0 1 │ Rx ≠ 0 │ Rd ADDRESS	16
X	SSO	EX Rd, LABSSO (Rx) 0 1 1 0 1 1 0 1 │ Rx ≠ 0 │ Rd 0 │ SEGMENT │ OFFSET	16
X	SLO	EX Rd, LABEL (Rx) 0 1 1 0 1 1 0 1 │ Rx ≠ 0 │ Rd 1 │ SEGMENT │ ▨ OFFSET	19

Operation

src<0:15>↔Rd<0:15>

Description

The contents of the source operand word are exchanged with the contents of the destination operand word. The destination operand is always a general-purpose word register designated by the Rd field of the instruction. The source operand is determined by the appropriate addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose source register.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

EXB

EXCHANGE source byte with destination byte

EXB

EXB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	EXB Rbd, Rbs 1 0 1 0 1 1 0 0 | Rbs | Rbd	6
IR	NS	EXB Rbd, Rs↑ 0 0 1 0 1 1 0 0 | Rs | Rbd	12
IR	S	EXB Rbd, RRs↑ 0 0 1 0 1 1 0 0 | RRs | Rbd	12
DA	NS	EXB Rbd, LABEL 0 1 1 0 1 1 0 0 | 0 0 0 0 | Rbd ADDRESS	15
DA	SSO	EXB Rbd, LABSSO 0 1 1 0 1 1 0 0 | 0 0 0 0 | Rbd 0 | SEGMENT | OFFSET	16
DA	SLO	EXB Rbd, LABEL 0 1 1 0 1 1 0 0 | 0 0 0 0 | Rbd 1 | SEGMENT | ▨ OFFSET	18
X	NS	EXB Rbd, LABEL (Rx) 0 1 1 0 1 1 0 0 | Rx ≠ 0 | Rbd ADDRESS	16
X	SSO	EXB Rbd, LABSSO (Rx) 0 1 1 0 1 1 0 0 | Rx ≠ 0 | Rbd 0 | SEGMENT | OFFSET	16
X	SLO	EXB Rbd, LABEL (Rx) 0 1 1 0 1 1 0 0 | Rx ≠ 0 | Rbd 1 | SEGMENT | ▨ OFFSET	19

Operation

src<0:7>↔Rbd<0:7>

Description

The contents of the source operand byte are exchanged with the contents of the destination operand byte. The destination operand is always a general-purpose byte register designated by the Rbd field of the instruction. The source operand is determined by the appropriate addressing mode.

In the IR mode R0 (or RR0) can be designated as the general-purpose source register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

EXTEND sign of a word

EXTS RRd

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	EXTS RRd 1 0 1 1 0 0 0 1 · RRd · 1 0 1 0	11	If RRd<0:15>is negative, RRd<16:31>←1's; otherwise RRd<16:31>←0.

Description

The destination is a general-purpose register pair, designated by the RRd field of the instruction. The sign bit of the less significant register of the pair is copied into each bit position of the most significant register. In this manner, the sign of the operand is preserved as the operand is extended from 16 to 32 bits in length.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

EXTSB

EXTEND sign of a byte

EXTSB

EXTSB Rd

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	EXTSB Rd 1 0 1 1 0 0 0 1 Rd 0 0 0 0	11

Operation

If Rd<0:7>is negative,
Rd<8:15>←1's;
otherwise Rd<8:15>←0.

Description

The destination is a general-purpose register, designated by the Rd field of the instruction. The sign bit of the less significant byte of the register is copied into each position of the most significant byte. In this manner, the sign of the operand is preserved as the operand is extended from eight to 16 bits.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

EXTSL EXTSL

EXTEND sign of a long word

EXTSL RQd

Mode	Version	Mnemonic and Form	Clocks
		EXTSL RQd	
R	NS, S	1 0 1 1 0 0 0 1 | RQd | 0 1 1 1	11

Operation

If RQd<0:31>is negative,
RQd<32:63>←1's;
otherwise RQd<32:63>←0.

Description

The destination is a general-purpose register quad, designated by the RQd field of the instruction. The sign bit of the less significant register pair of the quad is copied into each bit position of the most significant register pair. In this manner, the sign of the operand is preserved as the operand is extended from 32 to 64 bits.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Flags are not affected.

HALT

HALT

HALT

HALT

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
		HALT	
R	NS, S	0111 1010 0000 0000	8 + 3n*

*Interrupts are recognized at the end of each three cycle period.

Operation

Description

Instruction execution is suspended and CPU will be in a halted state until an interrupt or reset is received.

While in the halted state, bus requests will be acknowledged and memory refresh will continue.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

IN

INPUT word to register from I/O port

IN

IN Rd, src

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
PR	NS, S	IN Rd, Rp 0 0 1 1 1 1 0 1 Rp Rd	10
PA	NS, S	IN Rd, PORT 0 0 1 1 1 0 1 1 Rd 0 1 0 0 PORT ADDRESS	12

Operation

Rd<0:7>←port src<0:7>

Description

A general-purpose byte destination register designated by the Rd field of the instruction is loaded from an input port. The port address source is determined by the applicable port addressing mode. The original contents of the destination are lost.

R0 can be designated as the general-purpose port source register.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Flags are not affected.

INB **INPUT** byte to register from I/O port **INB**

IN Rbd, src

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
PR	NS, S	IN Rbd, Rp 0 0 1 1 1 1 0 0 | Rp | Rbd	10
PA	NS, S	IN Rbd, PORT 0 0 1 1 1 0 1 0 | Rbd | 0 1 0 0 PORT ADDRESS	12

Operation

Rbd<0:7>←port src<0:7>

Description

A general-purpose byte destination register designated by the Rbd field of the instruction is loaded from an input port. The port address source is determined by the applicable port addressing mode. The original contents of the destination are lost.

R0 can be designated as the general-purpose port source register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

INC

INCREMENT word

INC

INC dst, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	INC Rd, N 1 0 1 0 1 0 0 1 | Rd | n	4
IR	NS	INC Rd↑, N 0 0 1 0 1 0 0 1 | Rd | n	11
IR	S	INC RRd↑, N 0 0 1 0 1 0 0 1 | RRd | n	11
DA	NS	INC LABEL, N 0 1 1 0 1 0 0 1 | 0 0 0 0 | n ADDRESS	13
DA	SSO	INC LABSSO, N 0 1 1 0 1 0 0 1 | 0 0 0 0 | n 0 | SEGMENT | OFFSET	14
DA	SLO	INC LABEL, N 0 1 1 0 1 0 0 1 | 0 0 0 0 | n 1 | SEGMENT | ▨ OFFSET	16
X	NS	INC LABEL (Rx), N 0 1 1 0 1 0 0 1 | Rx ≠ 0 | n ADDRESS	14
X	SSO	INC LABSSO (Rx), N 0 1 1 0 1 0 0 1 | Rx ≠ 0 | n 0 | SEGMENT | OFFSET	14
X	SLO	INC LABEL (Rx), N 0 1 1 0 1 0 0 1 | Rx ≠ 0 | n 1 | SEGMENT | ▨ OFFSET	17

Operation

dst<0:15>←dst<0:15>+N

Description

A value between 1 and 16 is added to the destination operand word and the result is loaded back into the destination. The desired value to be added is specified by the n field, where n = zero corresponds to value one and so on, and n = F corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is one to 16, and n = N − 1. Specifying an N outside of the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
–	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

INCB

INCREMENT byte

INCB

INCB dst, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	INCB Rbd, N 1 0 1 0 1 0 0 0 Rbd n	4
IR	NS	INCB Rd↑, N 0 0 1 0 1 0 0 0 Rd n	11
IR	S	INCB RRd↑, N 0 0 1 0 1 0 0 0 RRd n	11
DA	NS	INCB LABEL, N 0 1 1 0 1 0 0 0 0 0 0 0 n ADDRESS	13
DA	SSO	INCB LABSSO, N 0 1 1 0 1 0 0 0 0 0 0 0 n 0 SEGMENT OFFSET	14
DA	SLO	INCB LABEL, N 0 1 1 0 1 0 0 0 0 0 0 0 n 1 SEGMENT ▨ OFFSET	16
X	NS	INCB LABEL (Rx), N 0 1 1 0 1 0 0 0 Rx ≠ 0 n ADDRESS	14
X	SSO	INCB LABSSO (Rx), N 0 1 1 0 1 0 0 0 Rx ≠ 0 n 0 SEGMENT OFFSET	14
X	SLO	INCB LABEL (Rx), N 0 1 1 0 1 0 0 0 Rx ≠ 0 n 1 SEGMENT ▨ OFFSET	17

Operation

dst<0:7>←dst<0:7>+N

Description

A value between 1 and 16 is added to the destination operand byte and the result is loaded back into the destination. The desired value to be added is specified by the n field, where n = zero corresponds to one and so on, and n = F corresponds to value 16. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.

In the IR mode R0 (or RR0) can be designated as the general-purpose destination register.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is one to 16, and n = N − 1. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	*	*	*	0	*

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.
DA: Set to 0.
H: Set on carry from least significant digit. Reset otherwise.

INPUT word from I/O port to memory, autodecrement

IND dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	IND Rd↑, Rp, Rc 0 0 1 1 1 0 1 1 · Rp · 1 0 0 0 0 0 0 0 · Rc · Rd · 1 0 0 0	21	dst<0:15>←port src<0:15> Rd<0:15>←Rd<0:15>−2 Rc<0:15>←Rc<0:15>−1
IR, PR	S	IND RRd↑, Rp, Rc 0 0 1 1 1 0 1 1 · Rp · 1 0 0 0 0 0 0 0 · Rc · RRd · 1 0 0 0	21	

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

INPUT byte from I/O port to memory, autodecrement

INDB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	INDB Rd↑, Rp, Rc 0 0 1 1 1 0 1 0 | Rp | 1 0 0 0 0 0 0 0 | Rc | Rd | 1 0 0 0	21	dst<0:7>←port src<0:7> Rd<0:15>←Rd<0:15>−1 Rc<0:15>←Rc<0:15>−1
IR, PR	S	INDB RRd↑, Rp, Rc 0 0 1 1 1 0 1 0 | Rp | 1 0 0 0 0 0 0 0 | Rc | RRd | 1 0 0 0	21	

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd and Rc are then decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	✱	–	–

P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
✱ = Conditional – see description

INPUT word from I/O port to memory, autodecrement and repeat

INDR dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		INDR Rd↑, Rp, Rc				
IR, PR	NS	0 0 1 1	1 0 1 1	Rp	1 0 0 0	11 + 10n*
		0 0 0 0	Rc	Rd	0 0 0 0	
		INDR RRd↑, Rp, Rc				
IR, PR	S	0 0 1 1	1 0 1 1	Rp	1 0 0 0	11 + 10n*
		0 0 0 0	Rc	RRd	0 0 0 0	

*n is the number of iterations.

Operation

dst<0:15>←port src<0:15>
Rd<0:15>←Rd<0:15>−2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

INDRB

INPUT byte from I/O port to memory, autodecrement and repeat

INDRB

INDRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		INDRB Rd↑, Rp, Rc				
IR, PR	NS	0,0,1,1,1,0,1,0		Rp	1,0,0,0	11 + 10n*
		0,0,0,0	Rc	Rd	0,0,0,0	
		INDRB RRd↑, Rp, Rc				
IR, PR	S	0,0,1,1,1,0,1,0		Rp	1,0,0,0	11 + 10n*
		0,0,0,0	Rc	RRd	0,0,0,0	

*n is the number of iterations.

Operation

dst<0:7>← port src<0:7>
Rd<0:15>←Rd<0:15>−1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into the memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd is then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

INI

INPUT word from I/O port to memory, autoincrement

INI

INI dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		INI Rd↑, Rp, Rc				
IR, PR	NS	0,0,1,1	1,0,1,1	Rp	0,0,0,0	21
		0,0,0,0	Rc	Rd	1,0,0,0	
		INI RRd↑, Rp, Rc				
IR, PR	S	0,0,1,1	1,0,1,1	Rp	0,0,0,0	21
		0,0,0,0	Rc	RRd	1,0,0,0	

Operation

dst<0:15>←port src<0:15>
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	PV	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

INIB

INPUT byte from I/O port to memory, autoincrement

INIB

INIB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		INIB Rd↑, Rp, Rc				
IR, PR	NS	0 0 1 1	1 0 1 0	Rp	0 0 0 0	21
		0 0 0 0	Rc	Rd	1 0 0 0	
		INIB RRd↑, Rp, Rc				
IR, PR	S	0 0 1 1	1 0 1 0	Rp	0 0 0 0	21
		0 0 0 0	Rc	RRd	1 0 0 0	

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc register is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

INIR

INPUT word from I/O port to memory, autoincrement and repeat

INIR

INIR dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		INIR Rd↑, Rp, Rc				
IR, PR	NS	0,0,1,1,1,0,1,1		Rp	0,0,0,0	11 + 10n*
		0,0,0,0	Rc	Rd	0,0,0,0	
		INIR RRd↑, Rp, Rc				
IR, PR	S	0,0,1,1,1,0,1,1		Rp	0,0,0,0	11 + 10n*
		0,0,0,0	Rc	RRd	0,0,0,0	

*n is the number of iterations.

Operation

dst<0:15>←port src<0:15>
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

The instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

INIRB — INPUT byte from I/O port to memory, autoincrement and repeat — INIRB

INIRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	INIRB Rd↑, Rp, Rc 0 0 1 1 1 0 1 0 · Rp · 0 0 0 0 0 0 0 0 · Rc · Rd · 0 0 0 0	11 + 10n*
IR, PR	S	INIRB RRd↑, Rp, Rc 0 0 1 1 1 0 1 0 · Rp · 0 0 0 0 0 0 0 0 · Rc · RRd · 0 0 0 0	11 + 10n*

*n is the number of iterations.

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

RETURN from interrupt

IRET

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
		IRET	
—	NS, S	0 1 1 1 1 0 1 1 | 0 0 0 0 | 0 0 0 0	13, 16

Operation

Nonsegmented

R15′<0:15>←R15′<0:15>+2
FCW←(R15<0:15>)
R15′<0:15>←R15′<0:15>+2
PC←(R15′<0:15>)
R15′<0:15>←R15′<0:15>+2

Segmented

R15′<0:15>←R15′<0:15>+2
FCW←(RR14′<0:22>)
R15′<0:15>←R15′<0:15>+2
PC SEGMENT←(RR14′<0:22>)
R15′<0:15>←R15′<0:15>+2
PC OFFSET←(RR14′<0:22>)
R15′<0:15>←R15′<0:15>+2

Description

This instruction causes a return from an interrupt or trap. The program status that was pushed on the system stack is popped to restore the pre-interrupt processor status. The System Stack Pointer contents are modified to reflect the entries that are removed.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	∗	∗

The flags will be restored to pre-interrupt values.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

JUMP conditional

JP cc, dst

Mode	Version	Mnemonic and Form	Clocks CC True/CC False
IR	NS	JP CC, Rd↑ 0 0 0 1 1 1 1 0 | Rd | CC	7 / 7
IR	S	JP CC, RRd↑ 0 0 0 1 1 1 1 0 | RRd | CC	15 / 10
DA	NS	JP CC, LABEL 0 1 0 1 1 1 1 0 | 0 0 0 0 | CC ADDRESS	7 / 7
DA	SSO	JP CC, LABSSO 0 1 0 1 1 1 1 0 | 0 0 0 0 | CC 0 | SEGMENT | OFFSET	8 / 8
DA	SLO	JP CC, LABEL 0 1 0 1 1 1 1 0 | 0 0 0 0 | CC 1 | SEGMENT | ▨ OFFSET	10 / 10
X	NS	JP CC, LABEL (Rx) 0 1 0 1 1 1 1 0 | Rx ≠ 0 | CC ADDRESS	8 / 8
X	SSO	JP CC, LABSSO (Rx) 0 1 0 1 1 1 1 0 | Rx ≠ 0 | CC 0 | SEGMENT | OFFSET	8 / 8
X	SLO	JP CC, LABEL (Rx) 0 1 0 1 1 1 1 0 | Rx ≠ 0 | CC 1 | SEGMENT | ▨ OFFSET	11 / 11

Operation

PC<0:15>←dst<0:15>
if condition is met.
PC<0:15>←Updated PC
if condition failed.

Description

The program executes a jump if the condition set in the condition code field of the instruction is met. A destination address is obtained according to the applicable addressing mode, and is loaded into the program counter. If the condition set in the condition code of the instruction is not met, then the value in the program counter is merely updated.

R0 can be designated as the general-purpose destination register Rd or RRd in the IR mode.

Assembler Notation

Specifying condition CC is optional. If none is specified, the CC field of the instruction is set to hex eight.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

JR

JUMP conditional relative

JR

JR CC, LAB

Mode	Version	Mnemonic and Form	Clocks
RA	NS, S	JR CC, LAB 1 1 1 0 CC DISPLACEMENT	6

Operation

PC←Updated PC+2x displacement if condition is met; otherwise PC←Updated PC.

Description

A program jump is taken if the condition code set in the CC field of the instruction is met. If the condition is met, the contents of the program counter are updated (incremented by two) and added to the contents of the 8-bit displacement field of the instruction, after the latter has been sign extended and left shifted (word aligned). The result is then loaded into the program counter as the jump address. If the condition is failed, the program counter is merely incremented by two. The range of the jump is +127 to −128 words with respect to the updated PC. The program counter segment number remains unchanged.

Assembler Notation

The label LAB is an address which is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside of the allowable range produces an assembler error.

Specifying condition CC is optional. If none is specified, the CC field of the instruction is set to hex eight.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Flags are not affected.

LD

LDR

LOAD word register into memory

LD dst, Rs
LDR LAB, Rs

LD

LDR

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LD Rd↑, Rs 0 0 1 0 1 1 1 1 | Rd | Rs	8
IR	S	LD RRd↑, Rs 0 0 1 0 1 1 1 1 | RRd | Rs	8
DA	NS	LD LABEL, Rs 0 1 1 0 1 1 1 1 | 0 0 0 0 | Rs ADDRESS	11
DA	SSO	LD LABSSO, Rs 0 1 1 0 1 1 1 1 | 0 0 0 0 | Rs 0 | SEGMENT | OFFSET	12
DA	SLO	LD LABEL, Rs 0 1 1 0 1 1 1 1 | 0 0 0 0 | Rs 1 | SEGMENT | ▨ OFFSET	14
X	NS	LD LABEL (Rx), Rs 0 1 1 0 1 1 1 1 | Rx ≠ 0 | Rs ADDRESS	12
X	SSO	LD LABSSO (Rx), Rs 0 1 1 0 1 1 1 1 | Rx ≠ 0 | Rs 0 | SEGMENT | OFFSET	12
X	SLO	LD LABEL (Rx), Rs 0 1 1 0 1 1 1 1 | Rx ≠ 0 | Rs 1 | SEGMENT | ▨ OFFSET	15
RA	NS, S	LDR LAB, Rs 0 0 1 1 0 0 1 1 | 0 0 0 0 | Rs DISPLACEMENT	14
BA	NS, S (See note)	LD Rd↑ (D), Rs 0 0 1 1 0 0 1 1 | Rd ≠ 0 | Rs DISPLACEMENT	14
BX	NS, S (See note)	LD Rd↑ (Rx), Rs 0 1 1 1 0 0 1 1 | Rd ≠ 0 | Rs ▨ | Rx | ▨	14

Operation

dst<0:15>←Rs<0:15>

Description

The word contents of the source register are loaded into the word destination. The source operand is always a general-purpose word register designated by the Rs field of the instruction. The destination is determined by the applicable addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRd ≠ 0, as destination base address register.

Assembler Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768.

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LD

LDR

LOAD word into register

LD Rd, src
LDR Rd, LAB

LD

LDR

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	LD Rd, Rs 1 0 1 0 0 0 0 1 | Rs | Rd	3
IM	NS, S	LD Rd, IM 0 0 1 0 0 0 0 1 | 0 0 0 0 | Rd OPERAND	7
IR	NS	LD Rd, Rs↑ 0 0 1 0 0 0 0 1 | Rs ≠ 0 | Rd	7
IR	S	LD Rd, RRs↑ 0 0 1 0 0 0 0 1 | RRs ≠ 0 | Rd	7
DA	NS	LD Rd, LABEL 0 1 1 0 0 0 0 1 | 0 0 0 0 | Rd ADDRESS	9
DA	SSO	LD Rd, LABSSO 0 1 1 0 0 0 0 1 | 0 0 0 0 | Rd 0 | SEGMENT | OFFSET	10
DA	SLO	LD Rd, LABEL 0 1 1 0 0 0 0 1 | 0 0 0 0 | Rd 1 | SEGMENT | ▨ OFFSET	12
X	NS	LD Rd, LABEL (Rx) 0 1 1 0 0 0 0 1 | Rx ≠ 0 | Rd ADDRESS	10
X	SSO	LD Rd, LABSSO (Rx) 0 1 1 0 0 0 0 1 | Rx ≠ 0 | Rd 0 | SEGMENT | OFFSET	10
X	SLO	LD Rd, LABEL (Rx) 0 1 1 0 0 0 0 1 | Rx ≠ 0 | Rd 1 | SEGMENT | ▨ OFFSET	13
RA	NS, S	LDR Rd, LAB 0 0 1 1 0 0 0 1 | 0 0 0 0 | Rd DISPLACEMENT	14
BA	NS, S (See note)	LD Rd, Rs↑ (D) 0 0 1 1 0 0 0 1 | Rs ≠ 0 | Rd DISPLACEMENT	14
BX	NS, S (See note)	LD Rd, Rs↑ (Rx) 0 1 1 1 0 0 0 1 | Rs ≠ 0 | Rd ▨ | Rx | ▨	14

Operation

Rd<0:15>←src<0:15>

Description

The source operand word is loaded into the destination word register. The source operand is determined by the applicable addressing mode and the destination is always a general-purpose register designated by the Rd field of the instruction. The contents of the source operand are unaltered, and the original contents of the destination are lost.

Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRs ≠ 0, as source base address register.

Assembler Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768.

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LD | **LD**

LOAD IMMEDIATE word into memory

LD dst, IM

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LD Rd↑, IM 0,0,0,0,1,1,0,1 │ Rd │ 0,1,0,1 OPERAND	11
IR	S	LD RRd↑, IM 0,0,0,0,1,1,0,1 │ RRd │ 0,1,0,1 OPERAND	11
DA	NS	LD LABEL, IM 0,1,0,0,1,1,0,1 │ 0,0,0,0 │ 0,1,0,1 ADDRESS OPERAND	14
DA	SSO	LD LABSSO, IM 0,1,0,0,1,1,0,1 │ 0,0,0,0 │ 0,1,0,1 0 │ SEGMENT │ OFFSET OPERAND	15
DA	SLO	LD LABEL, IM 0,1,0,0,1,1,0,1 │ 0,0,0,0 │ 0,1,0,1 1 │ SEGMENT │ ▨ OFFSET OPERAND	17
X	NS	LD LABEL (Rx), IM 0,1,0,0,1,1,0,1 │ Rx ≠ 0 │ 0,1,0,1 ADDRESS OPERAND	15
X	SSO	LD LABSSO (Rx), IM 0,1,0,0,1,1,0,1 │ Rx ≠ 0 │ 0,1,0,1 0 │ SEGMENT │ OFFSET OPERAND	15
X	SLO	LD LABEL (Rx), IM 0,1,0,0,1,1,0,1 │ Rx ≠ 0 │ 0,1,0,1 1 │ SEGMENT │ ▨ OFFSET OPERAND	18

Operation

dst<0:15>←IM<0:15>

Description

The immediate word operand following the instruction in memory is loaded into the destination. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

LDA

LDAR

LOAD address to register

LDA dst, addr
LDAR dst, LAB

LDA

LDAR

Mode	Version	Mnemonic and Form	Clocks
DA	NS	LDA Rd, LABEL 0 1 1 1 0 1 1 0 0 0 0 0 Rd ADDRESS	12
DA	SSO	LDA RRd, LABSSO 0 1 1 1 0 1 1 0 0 0 0 0 RRd 0 SEGMENT OFFSET	13
DA	SLO	LDA RRd, LABEL 0 1 1 1 0 1 1 0 0 0 0 0 RRd 1 SEGMENT ▨ OFFSET	15
X	NS	LDA Rd, LABEL (Rx) 0 1 1 1 0 1 1 0 Rx ≠ 0 Rd ADDRESS	13
X	SSO	LDA RRd, LABSSO (Rx) 0 1 1 1 0 1 1 0 Rx ≠ 0 RRd 0 SEGMENT OFFSET	13
X	SLO	LDA RRd, LABEL (Rx) 0 1 1 1 0 1 1 0 Rx ≠ 0 RRd 1 SEGMENT ▨ OFFSET	16
RA	NS, S (See note)	LDAR Rd, LAB 0 0 1 1 0 1 0 0 0 0 0 0 Rd DISPLACEMENT	15
BA	NS, S (See note)	LDA Rd, Rs↑ (D) 0 0 1 1 0 1 0 0 Rs ≠ 0 Rd DISPLACEMENT	15
BX	NS, S (See note)	LDA Rd, Rs↑ (Rx) 0 1 1 1 0 1 0 0 Rs ≠ 0 Rd ▨ Rx ▨	15

Operation

dst←address of source

Description

The address of a source operand is determined from the applicable addressing mode. This address is then loaded into the general-purpose destination register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost.

Note: The destination will be a word operand in the non-segmented version and requires the designation of a general-purpose word register, Rd. The destination will be a long word operand in the segmented version and requires the designation of a general-purpose register pair, RRd.

In the BA and BX addressing modes, the source base address register will be a general-purpose register, Rs, in the non-segmented version, and a register pair, RRs, in the segmented version.

Assembier Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768.

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDAR

LOAD RELATIVE address to register

LDAR

LDAR dst, LAB

(See also LDA – Load address to register)

Mode	Version	Mnemonic and Form	Clocks
		LDAR Rd, LAB	
RA	NS	0 0 1 1 0 1 0 0 | 0 0 0 0 | Rd DISPLACEMENT	15
		LDAR RRd, LAB	
RA	S	0 0 1 1 0 1 0 0 | 0 0 0 0 | RRd DISPLACEMENT	15

Operation

dst←address of source

Description

The address of a source operand is determined by the RA addressing mode. This address is then loaded into the general-purpose destination register designated by the Rd or RRd field of the instruction. The original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

LDB

LDRB

LOAD byte register into memory

LDB dst, Rbs
LDRB LAB, Rbs

LDB

LDRB

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LDB Rd↑, Rbs 0 0 1 0 1 1 1 0 | Rd | Rbs	8
IR	S	LDB RRd↑, Rbs 0 0 1 0 1 1 1 0 | RRd | Rbs	8
DA	NS	LDB LABEL, Rbs 0 1 1 0 1 1 1 0 | 0 0 0 0 | Rbs ADDRESS	11
DA	SSO	LDB LABSSO, Rbs 0 1 1 0 1 1 1 0 | 0 0 0 0 | Rbs 0 | SEGMENT | OFFSET	12
DA	SLO	LDB LABEL, Rbs 0 1 1 0 1 1 1 0 | 0 0 0 0 | Rbs 1 | SEGMENT | ▨ OFFSET	14
X	NS	LDB LABEL (Rx), Rbs 0 1 1 0 1 1 1 0 | Rx ≠ 0 | Rbs ADDRESS	12
X	SSO	LDB LABSSO (Rx), Rbs 0 1 1 0 1 1 1 0 | Rd ≠ 0 | Rbs 0 | SEGMENT | OFFSET	12
X	SLO	LDB LABEL (Rx), Rbs 0 1 1 0 1 1 1 0 | Rd ≠ 0 | Rbs 1 | SEGMENT | ▨ OFFSET	15
RA	NS, S	LDRB LAB, Rbs 0 0 1 1 0 0 1 0 | 0 0 0 0 | Rbs DISPLACEMENT	14
BA	NS, S (See note)	LDB Rd↑ (D), Rbs 0 0 1 1 0 0 1 0 | Rd ≠ 0 | Rbs DISPLACEMENT	14
BX	NS, S (See note)	LDB Rd↑ (Rx), Rbs 0 1 1 1 0 0 1 0 | Rd ≠ 0 | Rbs ▨ | Rx | ▨	14

Operation

dst<0:7>←Rbs<0:7>

Description

The byte contents of the source register are loaded into the byte destination. The source operand is always a general-purpose byte register designated by the Rbs field of the instruction. The destination is determined by the applicable addressing mode. The contents of the source are unaltered and the original contents of the destination are lost.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRd ≠ 0, as destination base address register.

Assembler Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768.

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

5

LDB

LDRB

LOAD byte into register

LDB Rbd, src
LDRB Rbd, LAB

LDB

LDRB

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	LDB Rbd, Rbs 1 0 1 0 0 0 0 0 | Rbs | Rbd	3
IM	NS, S	LDB Rbd, IMb 0 0 1 0 0 0 0 0 | 0 0 0 0 | Rbd 7 OPERAND 0 | 7 OPERAND 0	7
IR	NS	LDB Rbd, Rs↑ 0 0 1 0 0 0 0 0 | Rs ≠ 0 | Rbd	7
IR	S	LDB Rbd, RRs↑ 0 0 1 0 0 0 0 0 | RRs ≠ 0 | Rbd	7
DA	NS	LDB Rbd, LABEL 0 1 1 0 0 0 0 0 | 0 0 0 0 | Rbd ADDRESS	9
DA	SSO	LDB Rbd, LABSSO 0 1 1 0 0 0 0 0 | 0 0 0 0 | Rbd 0 | SEGMENT | OFFSET	10
DA	SLO	LDB Rbd, LABEL 0 1 1 0 0 0 0 0 | 0 0 0 0 | Rbd 1 | SEGMENT | ▨ OFFSET	12
X	NS	LDB Rbd, LABEL (Rx) 0 1 1 0 0 0 0 0 | Rx ≠ 0 | Rbd ADDRESS	10
X	SSO	LDB Rbd, LABSSO (Rx) 0 1 1 0 0 0 0 0 | Rx ≠ 0 | Rbd 0 | SEGMENT | OFFSET	10
X	SLO	LDB Rbd, LABEL (Rx) 0 1 1 0 0 0 0 0 | Rx ≠ 0 | Rbd 1 | SEGMENT | ▨ OFFSET	13
RA	NS, S	LDRB Rbd, LAB 0 0 1 1 0 0 0 0 | 0 0 0 0 | Rbd DISPLACEMENT	14
BA	NS, S (See note)	LDB Rbd, Rs↑ (D) 0 0 1 1 0 0 0 0 | Rs ≠ 0 | Rbd DISPLACEMENT	14
BX	NS, S (See note)	LDB Rbd, Rs↑ (Rx) 0 1 1 1 0 0 0 0 | Rs ≠ 0 | Rbd ▨ | Rx | ▨	14

Operation

Rbd<0:7>←src<0:7>

Description

The source operand byte is loaded into the destination byte register. The source operand is determined by the applicable addressing mode and the destination is always a general-purpose byte register designated by the Rbd field of the instruction. The contents of the source operand are unaltered, and the original contents of the destination are lost.

Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRs ≠ 0, as source base address register.

Assembler Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768.

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set

0 = Cleared
∗ = Conditional – see description

LDB

LOAD IMMEDIATE byte into memory

LDB

LDB dst, IMb

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LDB Rd↑, IMb 0 0 0 0 1 1 0 0 | Rd | 0 1 0 1 7 OPERAND 0 | 7 OPERAND 0	11
IR	S	LDB RRd↑, IMb 0 0 0 0 1 1 0 0 | RRd | 0 1 0 1 7 OPERAND 0 | 7 OPERAND 0	11
DA	NS	LDB LABEL, IMb 0 1 0 0 1 1 0 0 | 0 0 0 0 | 0 1 0 1 ADDRESS 7 OPERAND 0 | 7 OPERAND 0	14
DA	SSO	LDB LABSSO, IMb 0 1 0 0 1 1 0 0 | 0 0 0 0 | 0 1 0 1 0 | SEGMENT | OFFSET 7 OPERAND 0 | 7 OPERAND 0	15
DA	SLO	LDB LABEL, IMb 0 1 0 0 1 1 0 0 | 0 0 0 0 | 0 1 0 1 1 | SEGMENT | ▨ OFFSET 7 OPERAND 0 | 7 OPERAND 0	17
X	NS	LDB LABEL (Rx), IMb 0 1 0 0 1 1 0 0 | Rx ≠ 0 | 0 1 0 1 ADDRESS 7 OPERAND 0 | 7 OPERAND 0	15
X	SSO	LDB LABSSO (Rx), IMb 0 1 0 0 1 1 0 0 | Rd ≠ 0 | 0 1 0 1 0 | SEGMENT | OFFSET 7 OPERAND 0 | 7 OPERAND 0	15
X	SLO	LDB LABEL (Rx), IMb 0 1 0 0 1 1 0 0 | Rx ≠ 0 | 0 1 0 1 1 | SEGMENT | ▨ OFFSET 7 OPERAND 0 | 7 OPERAND 0	18

Operation

dst<0:7>←IMb<0:7>

Description

The immediate byte operand following the instruction in memory is loaded into the destination. The destination is determined by the applicable addressing mode. The original contents of the destination are lost.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDB

LOAD IMMEDIATE byte into a register

LDB

LDB Rbd, IMb

(See also LDK – Load Immediate digit into a register)

Mode	Version	Mnemonic and Form	Clocks
		LDB Rbd, IMb	
R	NS, S	1 1 0 0 | Rbd | IMb	5

Operation

Rbd<0:7>←IMb<0:7>

Description

The immediate byte value in the instruction field, IMb, is loaded into the destination. The destination is always a general-purpose byte register designated by the Rbd field of the instruction.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDCTL

LOAD control register from a register

LDCTL

LDCTL CR, Rs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
		LDCTL CR, Rs		CR<0:15>←Rs<0:15>
R	NS, S	0 1 1 1 1 1 0 1 · Rs · 1 · CR	7	

Description

The CPU control register specified in the CR field of the instruction is loaded from the general-purpose source word register specified by the Rs field of the instruction. The original contents of the control register are lost.

The CR field decodes are shown below:

CR Field	Destination	CR Notation
0 0 0	–	–
0 0 1	–	–
0 1 0	FCW	FCW
0 1 1	Refresh register (bits 1 through 15)	REFRESH
1 0 0	NPSAP segment	PSAPSEG
1 0 1	NPSAP upper offset	PSAPOFF
1 1 0	R14 (normal stack pointer segment)	NSPSEG
1 1 1	R15 (normal stack pointer offset)	NSPOFF

(Note that the LDCTL instruction is not used to load to the system stack pointer.)

Assembler Notation

The assembler determines the CR field by the reserved word specified according to the table above.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are affected only if the FCW is selected as the destination.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

LDCTL

LOAD control register into a register

LDCTL

LDCTL Rd, CR

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
		LDCTL Rd, CR		Rd<0:15>←CR<0:15>
R	NS, S	0 1 1 1 1 1 0 1 | Rd | 0 | CR	7	

Description

The contents of the CPU control register specified in the CR field of the instruction are loaded into the general-purpose destination word register specified by the Rd field of the instruction. The original contents of the destination are lost. Where a control register word of less than 16 bits is loaded into the destination register, zeros are loaded into the unused bit positions.

The CR field decodes are shown below:

Field	Source	CR Notation
0 0 0	–	–
0 0 1	–	–
0 1 0	FCW	FCW
0 1 1	Refresh register (bits 1 through 8)	REFRESH
1 0 0	NPSAP segment	PSAPSEG
1 0 1	NPSAP upper offset	PSAPOFF
1 1 0	R14 (normal stack pointer segment)	NSPSEG
1 1 1	R15 (normal stack pointer offset)	NSPOFF

(Note that the LDCTL instruction is not used to load from the system stack pointer.)

Assembler Notation

The assembler determines the CR field by the reserved word specified according to the table above.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LOAD flag byte from a register

LDCTLB FLAGS, Rbs

Mode	Version	Mnemonic and Form	Clocks
		LDCTLB FLAGS, Rbs	
R	NS, S	1 0 0 0 1 1 0 0 | Rbs | 1 0 0 1	7

Operation

FCW<0:7>←Rbs<0:7>

Description

The flag byte of the FCW is loaded from a general-purpose byte source register designated by the Rbs field of the instruction. The previous contents of the flag register are lost.

Assembler Notation

The assembler notation for the flag byte of the FCW is the reserved word: FLAGS.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	∗	∗

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

Flags are affected as described above.

LDCTLB

LOAD flag byte into a register

LDCTLB

LDCTLB Rbd, FLAGS

Mode	Version	Mnemonic and Form	Clocks
		LDCTLB Rbd, FLAGS	
R	NS, S	1 0 0 0 1 1 0 0 | Rbd | 0 0 0 1	7

Operation

Rbd<0:7> ← FCW<0:7>

Description

The flag byte of the FCW is loaded into the general-purpose byte destination register designated by the Rbd field of the instruction. The previous contents of the destination register are lost.

Assembler Notation

The assembler notation for the flag byte of the FCW is the reserved word: FLAGS.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

LDD

LOAD memory word to memory, autodecrement

LDD

LDD dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks
		LDD Rd↑, Rs↑, Rc				
IR	NS	1 0 1 1	1 0 1 1	Rs	1 0 0 1	20
		0 0 0 0	Rc	Rd	1 0 0 0	
		LDD RRd↑, RRs↑, Rc				
IR	S	1 0 1 1	1 0 1 1	RRs	1 0 0 1	20
		0 0 0 0	Rc	RRd	1 0 0 0	

Operation

dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>−2
Rd<0:15>←Rd<0:15>−2
Rc<0:15>←Rc<0:15>−1

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by two.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

LDDB

LOAD memory byte to memory, autodecrement

LDDB

LDDB dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks
		LDDB Rd↑, Rs↑, Rc				
IR	NS	1 0 1 1 1 0 1 0		Rs	1 0 0 1	20
		0 0 0 0	Rc	Rd	1 0 0 0	
		LDDB RRd↑, RRs↑, Rc				
IR	S	1 0 1 1 1 0 1 0		RRs	1 0 0 1	20
		0 0 0 0	Rc	RRd	1 0 0 0	

Operation

dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15> − 1
Rd<0:15>←Rd<0:15> − 1
Rc<0:15>←Rc<0:15> − 1

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by one.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LOAD memory word to memory, autodecrement and repeat

LDDR dst, src, Rc

Mode	Version	Mnemonic and Form	Clocks	Operation
IR	NS	LDDR Rd↑, Rs↑, Rc 1011 1011 | Rs | 1001 0000 | Rc | Rd | 0000	11 + 9n*	dst<0:15>←src<0:15> Rs<0:15>←Rs<0:15>−2 Rd<0:15>←Rd<0:15>−2 Rc<0:15>←Rc<0:15>−1 Repeat until termination.
IR	S	LDDR RRd↑, RRs↑, Rc 1011 1011 | RRs | 1001 0000 | Rc | RRd | 0000	11 + 9n*	

*n is the number of iterations.

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by two and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDDRB

LOAD memory byte to memory, autodecrement and repeat

LDDRB

LDDRB dst, src, Rc

Mode	Version	Mnemonic and Form	Clocks
		LDDRB Rd↑, Rs↑, Rc	
IR	NS	1 0 1 1 1 0 1 0 | Rs | 1 0 0 1 0 0 0 0 | Rc | Rd | 0 0 0 0	11 + 9n*
		LDDRB RRd↑, RRs↑, Rc	
IR	S	1 0 1 1 1 0 1 0 | RRs | 1 0 0 1 0 0 0 0 | Rc | RRd | 0 0 0 0	11 + 9n*

*n is the number of iterations.

Operation

dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>−1
Rd<0:15>←Rd<0:15>−1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both decremented by one and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

LDI

LOAD memory word to memory, autoincrement

LDI

LDI dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks
		LDI Rd↑, Rs↑, Rc				
IR	NS	1 0 1 1 1 0 1 1		Rs	0 0 0 1	20
		0 0 0 0	Rc	Rd	1 0 0 0	
		LDI RRd↑, RRs↑, Rc				
IR	S	1 0 1 1 1 0 1 1		RRs	0 0 0 1	20
		0 0 0 0	Rc	RRd	1 0 0 0	

Operation

dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>+2
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by two.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDIB

LOAD memory byte to memory, autoincrement

LDIB

LDIB dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks	Operation
		LDIB Rd↑, Rs↑, Rc					dst<0:7>←src<0:7>
IR	NS	1 0 1 1	1 0 1 0	Rs	0 0 0 1	20	Rs<0:15>←Rs<0:15>+1
		0 0 0 0	Rc	Rd	1 0 0 0		Rd<0:15>←Rd<0:15>+1
							Rc<0:15>←Rc<0:15>−1
		LDIB RRd↑, RRs↑, Rc					
IR	S	1 0 1 1	1 0 1 0	RRs	0 0 0 1	20	
		0 0 0 0	Rc	RRd	1 0 0 0		

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by one.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	✱	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

LDIR

LOAD memory word to memory, autoincrement and repeat

LDIR

LDIR dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks
		LDIR Rd↑, Rs↑, Rc				
IR	NS	1 0 1 1 1 0 1 1		Rs	0 0 0 1	11 + 9n*
		0 0 0 0	Rc	Rd	0 0 0 0	
		LDIR RRd↑, RRs↑, Rc				
IR	S	1 0 1 1 1 0 1 1		RRs	0 0 0 1	11 + 9n*
		0 0 0 0	Rc	RRd	0 0 0 0	

*n is the number of iterations.

Operation

dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>+2
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source word operand is loaded into the word destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by two and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero. This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDIRB

LOAD memory byte to memory, autoincrement and repeat

LDIRB

LDIRB dst, src, Rc

Mode	Version	Mnemonic and Form	Clocks
		LDIRB Rd↑, Rs↑, Rc	
IR	NS	1 0 1 1 1 0 1 0 | Rs | 0 0 0 1 0 0 0 0 | Rc | Rd | 0 0 0 0	11 + 9n*
		LDIRB RRd↑, RRs↑, Rc	
IR	S	1 0 1 1 1 0 1 0 | RRs | 0 0 0 1 0 0 0 0 | Rc | RRd | 0 0 0 0	11 + 9n*

*n is the number of iterations.

Operation

dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>+1
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

The source byte operand is loaded into the byte destination. Both the source and destination operands are addressed by the general-purpose registers designated in the Rs and Rd (or RRs and RRd) fields of the instruction. The contents of the source are unaltered and the original destination contents are lost. The contents of the general-purpose register designated by the Rc field of the instruction are decremented by one. The contents of Rs and Rd are both incremented by one and the operation will repeat until termination.

Termination occurs when the contents of Rc are zero.

This instruction is interruptible.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

LDK

LOAD IMMEDIATE digit into a register

LDK

LDK Rd, IMd

(See also LDB – Load Immediate byte into a register)

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	LDK Rd, IMd 1 0 1 1 1 1 0 1 | Rd | IM	5	dst<0:3>←IMd<0:3> dst<4:15>←0

Description

The immediate digit value in the instruction field, IMd, is loaded into the least significant four bits of the destination. The destination is a general-purpose word register designated by the Rd field of the instruction. Following LDK the remaining bits of the destination register are cleared.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDL dst, RRs

LDRL LAB, RRs

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LDL Rd↑, RRs 0 0 0 1 1 1 0 1 Rd RRs	11
IR	S	LDL RRd↑, RRs 0 0 0 1 1 1 0 1 RRd RRs	11
DA	NS	LDL LABEL, RRs 0 1 0 1 1 1 0 1 0 0 0 0 RRs ADDRESS	14
DA	SSO	LDL LABSSO, RRs 0 1 0 1 1 1 0 1 0 0 0 0 RRs 0 SEGMENT OFFSET	15
DA	SLO	LDL LABEL, RRs 0 1 0 1 1 1 0 1 0 0 0 0 RRs 1 SEGMENT ▨ OFFSET	17
X	NS	LDL LABEL (Rx), RRs 0 1 0 1 1 1 0 1 Rx ≠ 0 RRs ADDRESS	15
X	SSO	LDL LABSSO (Rx), RRs 0 1 0 1 1 1 0 1 Rx ≠ 0 RRs 0 SEGMENT OFFSET	15
X	SLO	LDL LABEL (Rx), RRs 0 1 0 1 1 1 0 1 Rx ≠ 0 RRs 1 SEGMENT ▨ OFFSET	18
RA	NS, S	LDRL LAB, RRs 0 0 1 1 0 1 1 1 0 0 0 0 RRs DISPLACEMENT	17
BA	NS, S (See note)	LDL Rd↑ (D), RRs 0 0 1 1 0 1 1 1 Rd ≠ 0 RRs DISPLACEMENT	17
BX	NS, S (See note)	LDL Rd↑ (Rx), RRs 0 1 1 1 0 1 1 1 Rd ≠ 0 RRs ▨ Rx ▨	17

Operation

dst<0:31>←RRs<0:31>

Description

The long word contents of the source register are loaded into the long word destination. The source operand is always a general-purpose long word register pair designated by the RRs field of the instruction. The long word destination is determined from the applicable addressing mode. The contents of the source are unaffected and the original contents of the destination are lost.

In the mode R0 (or RR0) can be designated as the general-purpose destination register.

Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRd = 0, as destination base address register.

Assembler Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDL

LDRL

LOAD long word into register

LDL RRd, src

LDRL RRd, LAB

LDL

LDRL

Mode	Version	Mnemonic and Form			Clocks
R	NS,S	LDL RRd, RRs			5
		1,0,0,1,0,1,0,0	RRs	RRd	
IM	NS,S	LDL RRd, IMℓ			11
		0,0,0,1,0,1,0,0	0,0,0,0	RRd	
		31 OPERAND 16			
		15 OPERAND 0			
IR	NS	LDL RRd, Rs↑			11
		0,0,0,1,0,1,0,0	Rs ≠ 0	RRd	
IR	S	LDL RRd, RRs↑			11
		0,0,0,1,0,1,0,0	RRs ≠ 0	RRd	
DA	NS	LDL RRd, LABEL			12
		0,1,0,1,0,1,0,0	0,0,0,0	RRd	
		ADDRESS			
DA	SSO	LDL RRd, LABSSO			13
		0,1,0,1,0,1,0,0	0,0,0,0	RRd	
		0 SEGMENT	OFFSET		
DA	SLO	LDL RRd, LABEL			15
		0,1,0,1,0,1,0,0	0,0,0,0	RRd	
		1 SEGMENT	▨	▨	
		OFFSET			
X	NS	LDL RRd, LABEL (Rx)			13
		0,1,0,1,0,1,0,0	Rx ≠ 0	RRd	
		ADDRESS			
X	SSO	LDL RRd, LABSSO (Rx)			13
		0,1,0,1,0,1,0,0	Rx ≠ 0	RRd	
		0 SEGMENT	OFFSET		
X	SLO	LDL RRd, LABEL (Rx)			16
		0,1,0,1,0,1,0,0	Rx ≠ 0	RRd	
		1 SEGMENT	▨	▨	
		OFFSET			
RA	NS,S	LDRL RRd, LAB			17
		0,0,1,1,0,1,0,1	0,0,0,0	RRd	
		DISPLACEMENT			
BA	NS, S (See note)	LDL RRd, Rs↑ (D)			17
		0,0,1,1,0,1,0,1	Rs ≠ 0	RRd	
		DISPLACEMENT			
BX	NS, S (See note)	LDL RRd, Rs↑ (Rx)			17
		0,1,1,1,0,1,0,1	Rs ≠ 0	RRd	
		▨ Rx	▨	▨	

Operation

RRd<0:31>←src<0:31>

Description

The source operand long word is loaded into the destination long word register. The source operand is determined by the applicable addressing mode and the destination is always a general-purpose long word register pair designated by the RRd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.

Note: In the BA and BX addressing modes the segmented version requires the designation of a register pair, RRs ≠ 0, as source base address register.

Assembler Notation

In the RA addressing mode the assembled displacement is a signed two's complement number with a range of +32,767 to −32,768.

In the RA addressing mode the label LAB is used by the assembler to generate the displacement relative to the updated PC.

In the BA addressing mode the value D is an unsigned integer which is assembled into the binary displacement.

A LAB or D which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LDM

LOAD multiple registers into memory

LDM

LDM dst, Rs, N

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LDM Rd↑, Rs, N 0 0 0 1 1 1 0 0 | Rd ≠ 0 | 1 0 0 1 ▨ | Rs | ▨ | n	11 + 3N
IR	S	LDM RRd↑, Rs, N 0 0 0 1 1 1 0 0 | RRd ≠ 0 | 1 0 0 1 ▨ | Rs | ▨ | n	11 + 3N
DA	NS	LDM LABEL, Rs, N 0 1 0 1 1 1 0 0 | 0 0 0 0 | 1 0 0 1 ▨ | Rs | ▨ | n ADDRESS	14 + 3N
DA	SSO	LDM LABSSO, Rs, N 0 1 0 1 1 1 0 0 | 0 0 0 0 | 1 0 0 1 ▨ | Rs | ▨ | n 0 | SEGMENT | OFFSET	15 + 3N
DA	SLO	LDM LABEL, Rs, N 0 1 0 1 1 1 0 0 | 0 0 0 0 | 1 0 0 1 ▨ | Rs | ▨ | n 1 | SEGMENT | ▨ OFFSET	17 + 3N
X	NS	LDM LABEL (Rx), Rs, N 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 1 0 0 1 ▨ | Rs | ▨ | n ADDRESS	15 + 3N
X	SSO	LDM LABSSO (Rx), Rs, N 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 1 0 0 1 ▨ | Rs | ▨ | n 0 | SEGMENT | OFFSET	15 + 3N
X	SLO	LDM LABEL (Rx), Rs, N 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 1 0 0 1 ▨ | Rs | ▨ | n 1 | SEGMENT | ▨ OFFSET	18 + 3N

Operation

dst+n<0:15>←Rs+n<0:15>
Repeat for n = 0, 1, 2, . . ., 15.

Description

A specified number of general-purpose word registers are loaded into memory. Loading will take place into consecutive memory locations with ascending addresses. The first register to be saved is specified in the Rs field of the instruction and registers will be accessed in ascending order, with R0 following R15. The number of registers to be saved is specified in the n field of the instruction. A zero in this field represents one register, etc. The destination address is determined by the applicable addressing mode using the Rd or Rx field of the instruction. The first register will be saved at this address. Succeeding registers will be saved at successive memory locations. The contents of the general-purpose registers are not altered.

This instruction is not interruptible.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is 1 to 16, and n = N − 1. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

LDM

LOAD multiple registers from memory

LDM

LDM Rd, src, N

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LDM Rd, Rs↑, N 0 0 0 1 1 1 0 0 | Rs ≠ 0 | 0 0 0 1 ▨ | Rd | ▨ | n	11 + 3N
IR	S	LDM Rd, RRs↑, N 0 0 0 1 1 1 0 0 | RRs ≠ 0 | 0 0 0 1 ▨ | Rd | ▨ | n	11 + 3N
DA	NS	LDM Rd, LABEL, N 0 1 0 1 1 1 0 0 | 0 0 0 0 | 0 0 0 1 ▨ | Rd | ▨ | n ADDRESS	14 + 3N
DA	SSO	LDM Rd, LABSSO, N 0 1 0 1 1 1 0 0 | 0 0 0 0 | 0 0 0 1 ▨ | Rd | ▨ | n 0 | SEGMENT | OFFSET	15 + 3N
DA	SLO	LDM Rd, LABEL, N 0 1 0 1 1 1 0 0 | 0 0 0 0 | 0 0 0 1 ▨ | Rd | ▨ | n 1 | SEGMENT | ▨ OFFSET	17 + 3N
X	NS	LDM Rd, LABEL (Rx), N 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 0 0 0 1 ▨ | Rd | ▨ | n ADDRESS	15 + 3N
X	SSO	LDM Rd, LABSSO (Rx), N 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 0 0 0 1 ▨ | Rd | ▨ | n 0 | SEGMENT | OFFSET	15 + 3N
X	SLO	LDM Rd, LABEL (Rx), N 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 0 0 0 1 ▨ | Rd | ▨ | n 1 | SEGMENT | ▨ OFFSET	18 + 3N

Operation

Rd+n<0:15>←src+n<0:15>
Repeat for n = 0, 1, 2, . . ., 15.

Description

A specified number of general-purpose word registers are loaded with words from consecutive memory locations with ascending addresses. The first register to be loaded is specified in the Rd field of the instruction. The registers will be addressed in ascending order for loading, with R0 following R15. The number of registers to be loaded is specified in the 'n' field of the instruction. A zero in this field represents one register, etc. A source operand address is generated according to the applicable addressing mode. The first register will be loaded from this location. Succeeding registers will be loaded from successive memory locations. The memory contents are not altered.

This instruction is not interruptible.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is 1 to 16, and n = N − 1. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LOAD program status

LDPS src

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR	NS	LDPS Rs↑ 0,0,1,1,1,0,0,1 Rs ≠ 0 0,0,0,0	12
IR	S	LDPS RRs↑ 0,0,1,1,1,0,0,1 RRs ≠ 0 0,0,0,0	16
DA	NS	LDPS LABEL 0,1,1,1,1,0,0,1 0,0,0,0 0,0,0,0 ADDRESS	16
DA	SSO	LDPS LABSSO 0,1,1,1,1,0,0,1 0,0,0,0 0,0,0,0 0 SEGMENT OFFSET	20
DA	SLO	LDPS LABEL 0,1,1,1,1,0,0,1 0,0,0,0 0,0,0,0 1 SEGMENT ▨ OFFSET	22
X	NS	LDPS LABEL (Rx) 0,1,1,1,1,0,0,1 Rx ≠ 0 0,0,0,0 ADDRESS	17
X	SSO	LDPS LABSSO (Rx) 0,1,1,1,1,0,0,1 Rx ≠ 0 0,0,0,0 0 SEGMENT OFFSET	20
X	SLO	LDPS LABEL (Rx) 0,1,1,1,1,0,0,1 Rx ≠ 0 0,0,0,0 1 SEGMENT ▨ OFFSET	23

Operation

Description

This instruction loads the processor status from consecutive memory locations with ascending address. The starting address of the status is determined by the applicable addressing mode. In AmZ8001 the status is four consecutive words, and in AmZ8002 the status is two words.

The PC segment number is not affected by this instruction in non-segmented mode.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	∗	∗

The processor flags are loaded with the contents of the new FCW.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LOAD RELATIVE word into register

LD Rd, LAB

(See also LD − Load word into register)

Mode	Version	Mnemonic and Form	Clocks
RA	NS,S	LDR Rd, LAB 0 0 1 1 0 0 0 1 ‖ 0 0 0 0 ‖ Rd DISPLACEMENT	14

Operation

Rd<0:15>←src<0:15>

Description

The source operand word is loaded into the destination word register. The source operand is determined by the RA addressing mode and the destination is a general-purpose word register designated by the Rd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
−	−	−	−	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

Flags are not affected.

LDR

LOAD RELATIVE word register into memory

LDR

LDR LAB, Rs

(See also LD – Load word register into memory)

Mode	Version	Mnemonic and Form	Clocks
RA	NS,S	LDR LAB, Rs	14

0	0	1	1	0	0	1	1	0	0	0	0	Rs
DISPLACEMENT												

Operation

dst<0:15>←Rs<0:15>

Description

The word contents of the source register are loaded into the word destination. The source operand is a general-purpose word register designated by the Rs field of the instruction. The destination is determined by the RA addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement relative to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

LDRB

LOAD RELATIVE byte register into memory

LDRB

LDRB LAB, Rbs

(See also LDB – Load byte register into memory)

Mode	Version	Mnemonic and Form	Clocks	Operation
RA	NS, S	LDRB LAB, Rbs 0 0 1 1 0 0 1 0 | 0 0 0 0 | Rbs DISPLACEMENT	14	dst<0:7>←Rbs<0:7>

Description

The byte contents of the source register are loaded into the byte destination. The source operand is a general-purpose byte register designated by the Rbs field of the instruction. The destination is determined by the RA addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

5

LDRB

LOAD RELATIVE byte into register

LDRB

LDRB Rbd, LAB

(See also LDB – Load byte into register)

Mode	Version	Mnemonic and Form	Clocks	Operation
RA	NS, S	LDRB Rbd, LAB 0 0 1 1 0 0 0 0 | 0 0 0 0 | Rbd DISPLACEMENT	14	Rbd<0:7>←src<0:7>

Description

The source operand byte is loaded into the destination byte register. The source operand is determined by the RA addressing mode and the destination is a general-purpose byte register designated by the Rbd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = *Unaffected*
1 = Set
0 = Cleared
✱ = Conditional – see description

LDRL

LOAD RELATIVE long word register into memory

LDRL

LDRL LAB, RRs

(See also LDL – LOAD long word register into memory)

Mode	Version	Mnemonic and Form	Clocks
		LDRL LAB, RRs	
RA	NS, S	0 0 1 1 0 1 1 1 | 0 0 0 0 | RRs DISPLACEMENT	17

Operation

dst<0:31>←RRs<0:31>

Description

The long word contents of the source register are loaded into the long word destination. The source operand is a general-purpose long word register designated by the RRs field of the instruction. The destination is determined by the RA addressing mode. The contents of the source are unaltered, and the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

LDRL

LOAD RELATIVE long word into register

LDRL

LDRL RRd, LAB

Mode	Version	Mnemonic and Form	Clocks
RA	NS, S	LDRL RRd, LAB	17

0	0	1	1	0	1	0	1	0	0	0	0	RRd
DISPLACEMENT												

Operation

RRd<0:31>←src<0:31>

Description

The source operand long word is loaded into the destination long word register. The source operand is determined by the RA addressing mode and the destination is a general-purpose long word register designated by the RRd field of the instruction. The contents of the source operand are unaltered while the original contents of the destination are lost.

Assembler Notation

The assembled displacement is a signed two's complement number with a range of +32,767 to −32,768. The label LAB is used by the assembler to generate the displacement that is added to the updated PC. A LAB which results in a displacement outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

MBIT

MULTIMICRO TEST

MBIT

MBIT

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
		MBIT		S flag←$\overline{\mu I}$
–	NS, S	0 1 1 1 1 0 1 1 | 0 0 0 0 | 1 0 1 0	7	

Description

The multimicro input line $\overline{\mu I}$ is tested.

5

Flags

C	Z	S	P/V	DA	H
–	*	*	–	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

S: Set to 1 if $\overline{\mu I}$ is inactive. Reset otherwise.
Z: Undefined.

MULTIMICRO REQUEST

MREQ Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
		MREQ Rc	
—	NS, S	0 1 1 1 1 0 1 1 · Rc · 1 1 0 1	12 + 7n*

*n is the number of decrementations.

(n = 0 if initial state of $\overline{\mu I}$ was LOW.)

Operation

See description below.

Description

There is an external input called Micro-In ($\overline{\mu I}$) and an output called Micro-Out ($\overline{\mu O}$). The MREQ instruction tests the state of the $\overline{\mu I}$ input. If the $\overline{\mu I}$ input is LOW, the instruction terminates. If the $\overline{\mu I}$ input is HIGH, the $\overline{\mu O}$ output is activated and the general-purpose register designated by the Rc field of the instruction is decremented by one. The state of the $\overline{\mu I}$ line is tested, and the contents of Rc are repeatedly decremented until they reach zero. The instruction then terminates with the $\overline{\mu O}$ line LOW if $\overline{\mu I}$ is LOW, or with the $\overline{\mu O}$ line HIGH if $\overline{\mu I}$ is HIGH.

Flags

C	Z	S	P/V	DA	H
–	*	*	–	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

S	Z	
0	0	Instruction terminated after initial test of $\overline{\mu I}$.
0	1	Instruction terminated due to contents of Rc reaching zero with $\overline{\mu I}$ HIGH.
1	1	Instruction terminated due to contents of Rc reaching zero with $\overline{\mu I}$ LOW.

MULTIMICRO RESET

MRES

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
–	NS, S	MRES 0 1 1 1 1 0 1 1 | 0 0 0 0 | 1 0 0 1	5

Operation

$\overline{\mu O}$←HIGH

Description

The multimicro out line $\overline{\mu O}$ is reset HIGH.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

Flags are not affected.

MULTIMICRO SET

MSET

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
		MSET	
—	NS, S	0111 1011 0000 1000	5

Operation

$\overline{\mu O} \leftarrow$ LOW

Description

The multimicro out line $\overline{\mu O}$ is set LOW. Note that this operation performs an unconditional setting of the $\overline{\mu O}$ line, independent of the state of the multimicro in line $\overline{\mu I}$.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = *Set*
0 = Cleared
∗ = Conditional – see description

MULTIPLY register with word

MULT RRd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	MULT RRd, Rs 1 0 0 1 1 0 0 1 | Rs | RRd	70
IM	NS, S	MULT, RRd, IM 0 0 0 1 1 0 0 1 | 0 0 0 0 | RRd OPERAND	70
IR	NS	MULT RRd, Rs↑ 0 0 0 1 1 0 0 1 | Rs ≠ 0 | RRd	70
IR	S	MULT RRd, RRs↑ 0 0 0 1 1 0 0 1 | RRs ≠ 0 | RRd	70
DA	NS	MULT RRd, LABEL 0 1 0 1 1 0 0 1 | 0 0 0 0 | RRd ADDRESS	71
DA	SSO	MULT RRd, LABSSO 0 1 0 1 1 0 0 1 | 0 0 0 0 | RRd 0 | SEGMENT | OFFSET	72
DA	SLO	MULT RRd, LABEL 0 1 0 1 1 0 0 1 | 0 0 0 0 | RRd 1 | SEGMENT | ▨ OFFSET	74
X	NS	MULT RRd, LABEL (Rx) 0 1 0 1 1 0 0 1 | Rx ≠ 0 | RRd ADDRESS	72
X	SSO	MULT RRd, LABSSO (Rx) 0 1 0 1 1 0 0 1 | Rx ≠ 0 | RRd 0 | SEGMENT | OFFSET	72
X	SLO	MULT RRd, LABEL (Rx) 0 1 0 1 1 0 0 1 | Rx ≠ 0 | RRd 1 | SEGMENT | ▨ OFFSET	75

Operation

RRd<0:31>←RRd<0:15>xsrc<0:15>

Description

The least significant word of a destination register pair (multiplicand) is multiplied by the contents of a source word operand (multiplier). The result is loaded into the destination, which is a general-purpose register pair, designated by the RRd field of the instruction. The source operand is determined from the applicable addressing mode. Both the multiplicand and multiplier are treated as signed two's complement 16-bit integers. The original contents of the destination are lost. The source contents are unaltered.

5

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	0	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Set to 1 if product is less than -2^{15} or greater than/equal to 2^{15}. Reset otherwise.
Z: Set to 1 if product is zero. Reset otherwise.
S: Set to 1 if product is negative. Reset otherwise.
P/V: Reset.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	MULTL RQd, RRs 1 0 0 1 1 0 0 0 · RRs · RQd	282 + 7n*
IM	NS, S	MULTL RQd, IMℓ 0 0 0 1 1 0 0 0 · 0 0 0 0 · RQd 31 OPERAND 16 15 OPERAND 0	282 + 7n*
IR	NS	MULTL RQd, Rs↑ 0 0 0 1 1 0 0 0 · Rs ≠ 0 · RQd	282 + 7n*
IR	S	MULTL RQd, RRs↑ 0 0 0 1 1 0 0 0 · RRs ≠ 0 · RQd	282 + 7n*
DA	NS	MULTL RQd, LABEL 0 1 0 1 1 0 0 0 · 0 0 0 0 · RQd ADDRESS	283 + 7n*
DA	SSO	MULTL RQd, LABSSO 0 1 0 1 1 0 0 0 · 0 0 0 0 · RQd 0 · SEGMENT · OFFSET	284 + 7n*
DA	SLO	MULTL RQd, LABEL 0 1 0 1 1 0 0 0 · 0 0 0 0 · RQd 1 · SEGMENT · ▨ OFFSET	286 + 7n*
X	NS	MULTL RQd, LABEL (Rx) 0 1 0 1 1 0 0 0 · Rx ≠ 0 · RQd ADDRESS	284 + 7n*
X	SSO	MULTL RQd, LABSSO (Rx) 0 1 0 1 1 0 0 0 · Rx ≠ 0 · RQd 0 · SEGMENT · OFFSET	284 + 7n*
X	SLO	MULTL RQd, LABEL (Rx) 0 1 0 1 1 0 0 0 · Rx ≠ 0 · RQd 1 · SEGMENT · ▨ OFFSET	287 + 7n*

*n is the number of bits equal to one in the absolute value of the least significant half of the destination operand.

Operation

RQd<0:63>←RQd<0:31>xsrc<0:31>

Description

The least significant long word of a destination register quadruple (multiplicand) is multiplied by the contents of a source long word operand (multiplier). The result is loaded into the destination, which is a general-purpose register quadruple designated by the RQd field of the instruction. The source operand is determined from the applicable addressing mode. Both the multiplicand and multiplier are treated as signed two's complement 32-bit integers. The original contents of the destination are lost. The source contents are unaltered.

Flags

C	Z	S	P/V	DA	H
*	*	*	0	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Set to 1 if product is less than -2^{31}, or greater than/equal to 2^{31}. Reset otherwise.
Z: Set to 1 if product is zero. Reset otherwise.
S: Set to 1 if product is negative. Reset otherwise.
P/V: Reset.

NEG

NEGATE word

NEG

NEG dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	NEG Rd 1 0 0 0 1 1 0 1 Rd 0 0 1 0	7
IR	NS	NEG Rd↑ 0 0 0 0 1 1 0 1 Rd 0 0 1 0	12
IR	S	NEG RRd↑ 0 0 0 0 1 1 0 1 RRd 0 0 1 0	12
DA	NS	NEG LABEL 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 ADDRESS	15
DA	SSO	NEG LABSSO 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 SEGMENT OFFSET	16
DA	SLO	NEG LABEL 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 SEGMENT ▨ OFFSET	18
X	NS	NEG LABEL (Rx) 0 1 0 0 1 1 0 1 Rx ≠ 0 0 0 1 0 ADDRESS	16
X	SSO	NEG LABSSO (Rx) 0 1 0 0 1 1 0 1 Rx ≠ 0 0 0 1 0 0 SEGMENT OFFSET	16
X	SLO	NEG LABEL (Rx) 0 1 0 0 1 1 0 1 Rx ≠ 0 0 0 1 0 1 SEGMENT ▨ OFFSET	19

Operation

$dst<0:15> \leftarrow \overline{dst<0:15>}+1$

Description

The contents of the destination word operand are replaced by its two's complement. The destination operand is obtained by using the applicable addressing mode. The negation is achieved by complementing the destination operand and adding one. The original contents of the destination are lost.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Reset on carry from destination. Set to 1 otherwise.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Set to 1 if operand value is 8000 (HEX). Reset otherwise.

NEGB

NEGATE byte

NEGB dst

NEGB

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	NEGB Rbd 1 0 0 0 1 1 0 0 Rbd 0 0 1 0	7
IR	NS	NEGB Rd↑ 0 0 0 0 1 1 0 0 Rd 0 0 1 0	12
IR	S	NEGB RRd↑ 0 0 0 0 1 1 0 0 RRd 0 0 1 0	12
DA	NS	NEGB LABEL 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 ADDRESS	15
DA	SSO	NEGB LABSSO 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 SEGMENT OFFSET	16
DA	SLO	NEGB LABEL 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 SEGMENT ▨ OFFSET	18
X	NS	NEGB LABEL (Rx) 0 1 0 0 1 1 0 0 Rx ≠ 0 0 0 1 0 ADDRESS	16
X	SSO	NEGB LABSSO (Rx) 0 1 0 0 1 1 0 0 Rx ≠ 0 0 0 1 0 0 SEGMENT OFFSET	16
X	SLO	NEGB LABEL (Rx) 0 1 0 0 1 1 0 0 Rx ≠ 0 0 0 1 0 1 SEGMENT ▨ OFFSET	19

Operation

$dst<0:7> \leftarrow \overline{dst<0:7>}+1$

Description

The contents of the destination byte operand are replaced by its two's complement. The destination operand is obtained by using the applicable addressing mode. The negation is achieved by complementing the destination operand and adding one. The original contents of the destination are lost.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Reset on carry from destination. Set to 1 otherwise.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if the operand value is 80 (HEX). Reset otherwise.

NOP

NO OPERATION

NOP

NOP

NOP

Mode	Version	Mnemonic and Form	Clocks
—	NS, S	NOP 1 0 0 0 1 1 0 1 │ 0 0 0 0 │ 0 1 1 1	7

Operation

(see description below)

Description

No operation takes place and PC is incremented by two.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

OR word with register

OR Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	OR Rd, Rs 1 0 0 0 0 1 0 1 | Rs | Rd	4
IM	NS, S	OR Rd, IM 0 0 0 0 0 1 0 1 | 0 0 0 0 | Rd OPERAND	7
IR	NS	OR Rd, Rs↑ 0 0 0 0 0 1 0 1 | Rs ≠ 0 | Rd	7
IR	S	OR Rd, RRs↑ 0 0 0 0 0 1 0 1 | RRs ≠ 0 | Rd	7
DA	NS	OR Rd, LABEL 0 1 0 0 0 1 0 1 | 0 0 0 0 | Rd ADDRESS	9
DA	SSO	OR Rd, LABSSO 0 1 0 0 0 1 0 1 | 0 0 0 0 | Rd 0 | SEGMENT | OFFSET	10
DA	SLO	OR Rd, LABEL 0 1 0 0 0 1 0 1 | 0 0 0 0 | Rd 1 | SEGMENT | ▨ OFFSET	12
X	NS	OR Rd, LABEL (Rx) 0 1 0 0 0 1 0 1 | Rx ≠ 0 | Rd ADDRESS	10
X	SSO	OR Rd, LABSSO (Rx) 0 1 0 0 0 1 0 1 | Rx ≠ 0 | Rd 0 | SEGMENT | OFFSET	10
X	SLO	OR Rd, LABEL (Rx) 0 1 0 0 0 1 0 1 | Rx ≠ 0 | Rd 1 | SEGMENT | ▨ OFFSET	13

Operation

Rd<0:15>←src<0:15> V Rd<0:15>

Description

Logical OR operation is performed between corresponding bits of the source and destination words. The source operand is obtained using the applicable addressing mode and the destination is always a general-purpose register designated by the Rd field of the instruction. The 16-bit result is loaded into the destination. The source operand is not altered and original destination operand is lost.

Flags

C	Z	S	P/V	DA	H
–	∗	∗	–	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.

ORB

OR byte with register

ORB

ORB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	ORB Rbd, Rbs 1 0 0 0 0 1 0 0 ∣ Rbs ∣ Rbd	4
IM	NS, S	ORB Rbd, IMb 0 0 0 0 0 1 0 0 ∣ 0 0 0 0 ∣ Rbd 7 OPERAND 0 ∣ 7 OPERAND 0	7
IR	NS	ORB Rbd, Rs↑ 0 0 0 0 0 1 0 0 ∣ Rs ≠ 0 ∣ Rbd	7
IR	S	ORB Rbd, RRs↑ 0 0 0 0 0 1 0 0 ∣ RRs ≠ 0 ∣ Rbd	7
DA	NS	ORB Rbd, LABEL 0 1 0 0 0 1 0 0 ∣ 0 0 0 0 ∣ Rbd ADDRESS	9
DA	SSO	ORB Rbd, LABSSO 0 1 0 0 0 1 0 0 ∣ 0 0 0 0 ∣ Rbd 0 ∣ SEGMENT ∣ OFFSET	10
DA	SLO	ORB Rbd, LABEL 0 1 0 0 0 1 0 0 ∣ 0 0 0 0 ∣ Rbd 1 ∣ SEGMENT ∣ ▨ OFFSET	12
X	NS	ORB Rbd, LABEL (Rx) 0 1 0 0 0 1 0 0 ∣ Rx ≠ 0 ∣ Rbd ADDRESS	10
X	SSO	ORB Rbd, LABSSO (Rx) 0 1 0 0 0 1 0 0 ∣ Rx ≠ 0 ∣ Rbd 0 ∣ SEGMENT ∣ OFFSET	10
X	SLO	ORB Rbd, LABEL (Rx) 0 1 0 0 0 1 0 0 ∣ Rx ≠ 0 ∣ Rbd 1 ∣ SEGMENT ∣ ▨ OFFSET	13

Operation

dst<0:7>←src<0:7> V dst<0:7>

Description

A logical OR operation is performed between corresponding bits of the source and destination bytes. The source byte is obtained using the applicable addressing mode and the destination byte is always a general-purpose byte register designated by the Rbd field of the instruction. The 8-bit result is loaded into the destination. The source byte is not altered and the original byte in the destination register is lost.

5

Flags

C	Z	S	P/V	DA	H
–	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if parity of result is even. Reset otherwise.

OTDR

OUTPUT word from memory to I/O port, autodecrement and repeat

OTDR

OTDR Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	OTDR Rp, Rs↑, Rc 0 0 1 1 1 0 1 1 │ Rs │ 1 0 1 0 0 0 0 0 │ Rc │ Rp │ 0 0 0 0	11 + 10n*
IR, PR	S	OTDR Rp, RRs↑, Rc 0 0 1 1 1 0 1 1 │ RRs │ 1 0 1 0 0 0 0 0 │ Rc │ Rp │ 0 0 0 0	11 + 10n*

*n is the number of iterations.

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>−2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

OTDRB **OTDRB**

OUTPUT byte to I/O port from memory, autodecrement and repeat

OTDRB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	OTDRB Rp, Rs↑, Rc 0 0 1 1 1 0 1 0 / Rs / 1 0 1 0 0 0 0 0 / Rc / Rp / 0 0 0 0	11 + 10n*	port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15> − 1 Rc<0:15>←Rc<0:15> − 1 Repeat until termination.
IR, PR	S	OTDRB Rp, RRs↑, Rc 0 0 1 1 1 0 1 0 / RRs / 1 0 1 0 0 0 0 0 / Rc / Rp / 0 0 0 0	11 + 10n*	

*n is the number of iterations.

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by Rc field are decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

OTIR

OUTPUT word to I/O port from memory, autoincrement and repeat

OTIR

OTIR Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		OTIR Rp, Rs↑, Rc				
IR, PR	NS	0 0 1 1 1 0 1 1		Rs	0 0 1 0	11 + 10n*
		0 0 0 0	Rc	Rp	0 0 0 0	
		OTIR Rp, RRs↑, Rc				
IR, PR	S	0 0 1 1 1 0 1 1		RRs	0 0 1 0	11 + 10n*
		0 0 0 0	Rc	Rp	0 0 0 0	

*n is the number of iterations.

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>+2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

OUTPUT byte to I/O port from memory, autoincrement and repeat

OTIRB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	OTIRB Rp, Rs↑, Rc 0,0,1,1,1,0,1,0 | Rs | 0,0,1,0 0,0,0,0 | Rc | Rp | 0,0,0,0	11 + 10n*
IR, PR	S	OTIRB Rp, RRs↑, Rc 0,0,1,1,1,0,1,0 | RRs | 0,0,1,0 0,0,0,0 | Rc | Rp | 0,0,0,0	11 + 10n*

*n is the number of iterations.

Operation

port dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one. This instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

OUTPUT word to I/O port from register

OUT dst, Rs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form			Clocks
PR	NS, S	OUT Rp, Rs			10
		0,0,1,1,1,1,1,1	Rp ≠ 0	Rs	
PA	NS, S	OUT PORT, Rs			12
		0,0,1,1,1,0,1,1	Rs	0,1,1,0	
		PORT ADDRESS			

Operation

port dst<0:15>←Rs<0:15>

Description

The contents of the general-purpose word source register designated by the Rs field of the instruction are loaded into an output port. The port address destination is determined by the applicable port addressing mode. The source contents are unaltered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

OUTB

OUTPUT byte to I/O port from register

OUTB

OUTB dst, Rbs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form			Clocks
		OUTB Rp, Rbs			
PR	NS	0 0 1 1 1 1 1 0	Rp ≠ 0	Rbs	10
		OUTB PORT, Rbs			
PA	NS, S	0 0 1 1 1 0 1 0	Rbs	0 1 1 0	12
		PORT ADDRESS			

Operation

port dst<0:7>←Rs<0:7>

Description

The contents of the general-purpose byte source register designated by the Rbs field of the instruction are loaded into an output port. The port address destination is determined by the applicable port addressing mode. The source contents are unaltered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

5

OUTPUT word to I/O port from memory, autodecrement

OUTD Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		OUTD Rp, Rs, Rc				
IR, PR	NS	0,0,1,1,1,0,1,1		Rs	1,0,1,0	21
		0,0,0,0	Rc	Rp	1,0,0,0	
		OUTD Rp, RRs, Rc				
IR, PR	S	0,0,1,1,1,0,1,1		RRs	1,0,1,0	21
		0,0,0,0	Rc	Rp	1,0,0,0	

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>−2
Rc<0:15>←Rc<0:15>−1

Description

Data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

OUTPUT byte to I/O port from memory, autodecrement

OUTDB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	OUTDB Rp, Rs↑, Rc 0,0,1,1,1,0,1,0 | Rs | 1,0,1,0 0,0,0,0 | Rc | Rp | 1,0,0,0	21	port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>−1 Rc<0:15>←Rc<0:15>−1
IR, PR	S	OUTDB Rp, RRs↑, Rc 0,0,1,1,1,0,1,0 | RRs | 1,0,1,0 0,0,0,0 | Rc | Rp | 1,0,0,0	21	

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

OUTI

OUTPUT word to I/O port from memory, autoincrement

OUTI

OUTI Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	OUTI Rp, Rs↑, Rc	21
IR, PR	S	OUTI Rp, RRs↑, Rc	21

OUTI Rp, Rs↑, Rc

0 0 1 1	1 0 1 1	Rs	0 0 1 0
0 0 0 0	Rc	Rp	1 0 0 0

OUTI Rp, RRs↑, Rc

0 0 1 1	1 0 1 1	RRs	0 0 1 0
0 0 0 0	Rc	Rp	1 0 0 0

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>+2
Rc<0:15>←Rc<0:15>−1

Description

A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

OUTPUT byte to I/O port from memory, autoincrement

OUTIB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	OUTIB Rp, Rs↑, Rc	21
IR, PR	S	OUTIB Rp, RRs↑, Rc	21

OUTIB Rp, Rs↑, Rc

0 0 1 1	1 0 1 0	Rs	0 0 1 0
0 0 0 0	Rc	Rp	1 0 0 0

OUTIB Rp, RRs↑, Rc

0 0 1 1	1 0 1 0	RRs	0 0 1 0
0 0 0 0	Rc	Rp	1 0 0 0

Operation

port dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

POP

POP word

POP

POP dst, Rs↑

Mode	Version	Mnemonic and Form			Clocks
R	NS, S	POP Rd, Rs↑			8
		1 0 0 1 0 1 1 1	Rs ≠ 0	Rd	
IR	NS	POP Rd↑, Rs↑			12
		0 0 0 1 0 1 1 1	Rs ≠ 0	Rd	
IR	S	POP RRd↑, Rs↑			12
		0 0 0 1 0 1 1 1	Rs ≠ 0	RRd	
DA	NS	POP LABEL, Rs↑			16
		0 1 0 1 0 1 1 1	Rs ≠ 0	0 0 0 0	
		ADDRESS			
DA	SSO	POP LABSSO, Rs↑			16
		0 1 0 1 0 1 1 1	Rs ≠ 0	0 0 0 0	
		0 SEGMENT	OFFSET		
DA	SLO	POP LABEL, Rs↑			18
		0 1 0 1 0 1 1 1	Rs ≠ 0	0 0 0 0	
		1 SEGMENT	▨		
		OFFSET			
X	NS	POP LABEL (Rx), Rs↑			16
		0 1 0 1 0 1 1 1	Rs ≠ 0	Rx ≠ 0	
		ADDRESS			
X	SSO	POP LABSSO (Rx), Rs↑			16
		0 1 0 1 0 1 1 1	Rs ≠ 0	Rx ≠ 0	
		0 SEGMENT	OFFSET		
X	SLO	POP LABEL (Rx), Rs↑			19
		0 1 0 1 0 1 1 1	Rs ≠ 0	Rx ≠ 0	
		1 SEGMENT	▨		
		OFFSET			

Operation

Description

The word from the memory location addressed by the general-purpose register designated by Rs, is loaded into the destination. The contents of the register designated by Rs are then automatically incremented by two. Thus, if the general-purpose register designated by Rs is regarded as a stack pointer, then the operation described above can be regarded as a POP. Any general-purpose register except R0 may be utilized as a stack pointer. The destination is determined by the applicable addressing mode.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

POPL

POP long word

POPL dst, Rs↑

POPL

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	POPL Rd, Rs↑ 1 0 0 1 0 1 0 1 | Rs ≠ 0 | Rd	12
IR	NS	POPL Rd↑, Rs↑ 0 0 0 1 0 1 0 1 | Rs ≠ 0 | Rd	19
IR	S	POPL RRd↑, Rs↑ 0 0 0 1 0 1 0 1 | Rs ≠ 0 | RRd	19
DA	NS	POPL LABEL, Rs↑ 0 1 0 1 0 1 0 1 | Rs ≠ 0 | 0 0 0 0 ADDRESS	22
DA	SSO	POPL LABSSO, Rs↑ 0 1 0 1 0 1 0 1 | Rs ≠ 0 | 0 0 0 0 0 | SEGMENT | OFFSET	23
DA	SLO	POPL LABEL, Rs↑ 0 1 0 1 0 1 0 1 | Rs ≠ 0 | 0 0 0 0 1 | SEGMENT | ▨ OFFSET	25
X	NS	POPL LABEL (Rx), Rs↑ 0 1 0 1 0 1 0 1 | Rs ≠ 0 | Rx ≠ 0 ADDRESS	23
X	SSO	POPL LABSSO (Rx), Rs↑ 0 1 0 1 0 1 0 1 | Rs ≠ 0 | Rx ≠ 0 0 | SEGMENT | OFFSET	23
X	SLO	POPL LABEL (Rx), Rs↑ 0 1 0 1 0 1 0 1 | Rs ≠ 0 | Rx ≠ 0 1 | SEGMENT | ▨ OFFSET	26

Operation

Description

The long word from the memory location addressed by the general-purpose register designated by Rs is loaded into the destination. The contents of the register designated by Rs are then automatically incremented by four. Thus, if the general-purpose register designated by Rs is regarded as a stack pointer, then the operation described above can be regarded as a POP. Any general-purpose register except R0 may be utilized as a stack-pointer. The destination operand is determined by the applicable addressing mode.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

PUSH

PUSH word

PUSH Rd↑, src

PUSH

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	PUSH Rd↑, Rs 1 0 0 1 0 0 1 1 Rd ≠ 0 Rs	9
IM	NS, S	PUSH Rd↑, IM 0 0 0 0 1 1 0 1 Rd ≠ 0 1 0 0 1 OPERAND	12
IR	NS	PUSH Rd↑, Rs↑ 0 0 0 1 0 0 1 1 Rd ≠ 0 Rs	13
IR	S	PUSH Rd↑, RRs↑ 0 0 0 1 0 0 1 1 Rd ≠ 0 RRs	13
DA	NS	PUSH Rd↑, LABEL 0 1 0 1 0 0 1 1 Rd ≠ 0 0 0 0 0 ADDRESS	14
DA	SSO	PUSH Rd↑, LABSSO 0 1 0 1 0 0 1 1 Rd ≠ 0 0 0 0 0 0 SEGMENT OFFSET	14
DA	SLO	PUSH Rd↑, LABEL 0 1 0 1 0 0 1 1 Rd ≠ 0 0 0 0 0 1 SEGMENT ▨ OFFSET	16
X	NS	PUSH Rd↑, LABEL (Rx) 0 1 0 1 0 0 1 1 Rd ≠ 0 Rx ≠ 0 ADDRESS	14
X	SSO	PUSH Rd↑, LABSSO (Rx) 0 1 0 1 0 0 1 1 Rd ≠ 0 Rx ≠ 0 0 SEGMENT OFFSET	14
X	SLO	PUSH Rd↑, LABEL (Rx) 0 1 0 1 0 0 1 1 Rd ≠ 0 Rx ≠ 0 1 SEGMENT ▨ OFFSET	17

Operation

Description

The contents of the register designated by the Rd field of the instruction are decremented by two. The source word operand is then loaded into the memory location addressed by the general-purpose register designated in the Rd field of the instruction. Thus, if the general-purpose register designated by Rd is regarded as a stack pointer, then the operation described above can be regarded as a PUSH. Any general-purpose register except R0 can be utilized as a stack pointer. The source operand is determined by the applicable addressing mode.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = *Set*
0 = Cleared
∗ = Conditional – see description

PUSH long word

PUSH Rd↑, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	PUSHL Rd↑, Rs 1 0 0 1 0 0 0 1 · Rd ≠ 0 · Rs	12
IR	NS	PUSHL Rd↑, Rs↑ 0 0 0 1 0 0 0 1 · Rd ≠ 0 · Rs	20
IR	S	PUSHL Rd↑, RRs↑ 0 0 0 1 0 0 0 1 · Rd ≠ 0 · RRs	20
DA	NS	PUSHL Rd↑, LABEL 0 1 0 1 0 0 0 1 · Rd ≠ 0 · 0 0 0 0 ADDRESS	20
DA	SSO	PUSHL Rd↑ LABSSO 0 1 0 1 0 0 0 1 · Rd ≠ 0 · 0 0 0 0 0 · SEGMENT · OFFSET	21
DA	SLO	PUSHL Rd↑, LABEL 0 1 0 1 0 0 0 1 · Rd ≠ 0 · 0 0 0 0 1 · SEGMENT · ▨ OFFSET	23
X	NS	PUSHL Rd↑, LABEL (Rx) 0 1 0 1 0 0 0 1 · Rd ≠ 0 · Rx ≠ 0 ADDRESS	21
X	SSO	PUSHL Rd↑, LABSSO (Rx) 0 1 0 1 0 0 0 1 · Rd ≠ 0 · Rx ≠ 0 0 · SEGMENT · OFFSET	21
X	SLO	PUSHL Rd↑, LABEL (Rx) 0 1 0 1 0 0 0 1 · Rd ≠ 0 · Rx ≠ 0 1 · SEGMENT · ▨ OFFSET	24

Operation

Description

The contents of the register designated by the Rd field of the instruction are decremented by four. The source long word operand is then loaded into the memory location addressed by the general-purpose register designated in the Rd field of the instruction. Thus, if the general-purpose register designated by Rd is regarded as a stack pointer, then the operation described above can be regarded as a PUSH. Any general-purpose register except R0 can be utilized as a stack pointer. The source operand is determined by the applicable addressing mode.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

RES

RESET bit in word (static)

RES

RES dst, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RST Rd, B 1 0 1 0 0 0 1 1 | Rd | b	4
IR	NS	RST Rd↑, B 0 0 1 0 0 0 1 1 | Rd ≠ 0 | b	11
IR	S	RST RRd↑, B 0 0 1 0 0 0 1 1 | RRd ≠ 0 | b	11
DA	NS	RST LABEL, B 0 1 1 0 0 0 1 1 | 0 0 0 0 | b ADDRESS	13
DA	SSO	RST LABSSO, B 0 1 1 0 0 0 1 1 | 0 0 0 0 | b 0 | SEGMENT | OFFSET	14
DA	SLO	RST LABEL, B 0 1 1 0 0 0 1 1 | 0 0 0 0 | b 1 | SEGMENT | ▨ OFFSET	16
X	NS	RST LABEL (Rx), B 0 1 1 0 0 0 1 1 | Rx ≠ 0 | b ADDRESS	14
X	SSO	RST LABSSO (Rx), B 0 1 1 0 0 0 1 1 | Rx ≠ 0 | b 0 | SEGMENT | OFFSET	14
X	SLO	RST LABEL (Rx), B 0 1 1 0 0 0 1 1 | Rx ≠ 0 | b 1 | SEGMENT | ▨ OFFSET	17

Operation

word dst<b bit><–0

Assembler Notation

The assembler notation B is a numerical expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

RES

RESET bit in word (dynamic)

RES

RES Rd, Rs

Mode	Version	Mnemonic and Form	Clocks
		RES Rd, Rs	
R	NS, S	0 0 1 0 0 0 1 1 | 0 0 0 0 | Rs ▨ | Rd | ▨ | ▨	10

Operation

Rd<bit specified in Rs(0:3)>←0

Description

The selected bit of the word destination is reset to zero. The destination word operand is the general-purpose register designated by the Rd field of the instruction. The bit of the destination register to be reset is determined by binary decode of the least significant four bits of a general-purpose word register designated by the Rs field of the instruction. The remaining 15 bits of the destination are unaltered.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Flags are not affected.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RESB Rbd, B 1 0 1 0 0 0 1 0 │ Rbd │ b	4
IR	NS	RESB Rd↑, B 0 0 1 0 0 0 1 0 │ Rd ≠ 0 │ b	11
IR	S	RESB RRd↑, B 0 0 1 0 0 0 1 0 │ RRd ≠ 0 │ b	11
DA	NS	RESB LABEL, B 0 1 1 0 0 0 1 0 │ 0 0 0 0 │ b ADDRESS	13
DA	SSO	RESB LABSSO, B 0 1 1 0 0 0 1 0 │ 0 0 0 0 │ b 0 │ SEGMENT │ OFFSET	14
DA	SLO	RESB LABEL, B 0 1 1 0 0 0 1 0 │ 0 0 0 0 │ b 1 │ SEGMENT │ ▨ OFFSET	16
X	NS	RESB LABEL (Rx), B 0 1 1 0 0 0 1 0 │ Rx ≠ 0 │ b ADDRESS	14
X	SSO	RESB LABSSO (Rx), B 0 1 1 0 0 0 1 0 │ Rx ≠ 0 │ b 0 │ SEGMENT │ OFFSET	14
X	SLO	RESB LABEL (Rx), B 0 1 1 0 0 0 1 0 │ Rx ≠ 0 │ b 1 │ SEGMENT │ ▨ OFFSET	17

Operation

byte dst<b bit>←0

Description

The selected bit of the byte destination is reset to zero. The remaining seven bits are unaltered. The destination is determined by the applicable addressing mode, while the bit to be reset is determined by the binary value of the b field of the instruction.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 7, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

RESB

RESET bit in byte (dynamic)

RESB

RESB Rbd, Rs

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RESB Rbd, Rs	10

0 0 1 0	0 0 1 0	0 0 0 0	Rs
▨	Rbd	▨	▨

Operation

Rbd<bit specified in Rs(0:2)>←0

Description

The selected bit of the byte destination is reset to zero. The destination byte operand is the general-purpose register designated by the Rbd field of the instruction. The bit of the destination register to be reset is determined by binary decode of the least significant three bits of a general-purpose word register designated by the Rs field of the instruction. The remaining seven bits of the destination are unaltered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

Flags are not affected.

5

RESFLG

RESET FLAGS

RESFLG LIST

RESFLG

Mode	Version	Mnemonic and Form	Clocks	Operation
		RESFLG LIST		FCW<C;Z;S;P/V>←0
—	NS, S	1 0 0 0 1 1 0 1 | C Z S PV | 0 0 1 1	7	(see description below)

Description

The CPU flags C, Z, S and P/V are reset or unaltered, according to the bit settings in the instruction field as described in the table below.

Instruction Bit	If = 0	If = 1	Assembler Notation
7	No effect	Reset C flag	CY
6	No effect	Reset Z flag	ZR
5	No effect	Reset S flag	SGN
4	No effect	Reset P/V flag	PV or OV

Assembler Notation

The assembler notation LIST refers to a list of any or all of the following reserved words, separated by commas: CY, ZR, SGN, PY or OV. Note that PY and OV affect the same flag.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

See above.

– = Unaffected
1 = *Set*
0 = Cleared
∗ = Conditional – see description

RET

RETURN conditional from subroutine

RET

RET CC

Mode	Version	Mnemonic and Form	Clocks
		RET CC	CC True/CC False
—	NS, S	1 0 0 1 1 1 1 0 | 0 0 0 0 | CC	10,13 7,7

Operation

Non-segmented
If CC condition met:
PC←(R15<0:15>)
R15<0:15>←R15<0:15>+2

Otherwise:
PC←PC+2

Segmented
If CC condition met:
PC segment←(RR14<0:22>)
R15<0:15>←R15<0:15>+2
PC OFFSET←(RR14<0:22>)
R15<0:15>←R15<0:15>+2

Otherwise:
PC OFFSET←PC OFFSET+2

Note: In the system mode R15' and RR14' are used instead of R15 and RR14, respectively.

Description

This instruction conditionally returns the CPU to the calling program. During a subroutine call the return address was automatically stacked. This return address is popped from the stack into the PC to effect the return. If the flags do not satisfy the conditions specified by the CC field, the PC is not loaded with the return address but merely updated to the following instruction. The stack pointer remains unaltered from its original value if there is no return.

Assembler Notation

Specifying condition CC is optional. If none is specified, the CC field of the instruction is set to hex eight.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

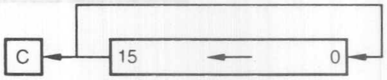
Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

RL Rd, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RL Rd, B 1 0 1 1 0 0 1 1 | Rd | 0 0 b 0	6 (one place) 7 (two places)

Operation



Description

The contents of the general-purpose word register designated by the Rd field of the instructions are rotated left. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from last bit rotated out of destination register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of destination register changed during rotation. Reset otherwise.

RLB

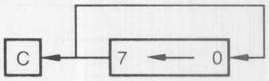
ROTATE byte left

RLB Rbd, B

RLB

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RLB Rbd, B 1 0 1 1 0 0 1 0 Rbd 0 0 b 0	6 (one place) 7 (two places)

Operation



Description

The contents of the general-purpose byte register designated by the Rbd field of the instruction are rotated left. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from last bit rotated out of destination register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of register changed during rotation. Reset otherwise.

RLC

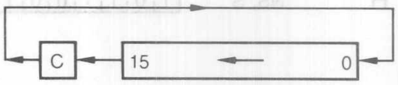
ROTATE word left through carry

RLC

RLC Rd, B

Mode	Version	Mnemonic and Form	Clocks
		RLC Rd, B	
R	NS, S	1 0 1 1 0 0 1 1 | Rbd | 1 0 b 0	6 (one place) 7 (two places)

Operation



Description

The contents of the destination word register designated by the Rd field of the instruction are rotated one or two places left. The last bit shifted out of the destination word is loaded into the carry flag, while the previous contents of the carry flag are shifted into the least significant bit of the destination word. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

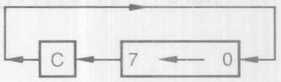
C: Loaded from the last bit rotated out of destination register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of destination contents changed during rotation. Reset otherwise.

ROTATE byte left through carry

RLCB Rbd, B

Mode	Version	Mnemonic and Form	Clocks
		RLCB Rbd, B	
R	NS, S	1 0 1 1 0 0 1 0 | Rd | 1 0 b 0	6 (one place) 7 (two places)

Operation



Description

The contents of the destination byte register designated by the Rbd field of the instruction are rotated one or two places left. The last bit shifted out of the destination byte is loaded into the carry flag, while the previous contents of the carry flag are rotated into the least significant bit of the destination byte. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	−	−

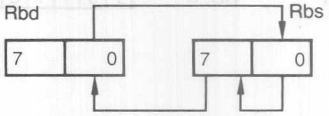
− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

C: Loaded from the last bit rotated out of destination.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of destination changed during rotation. Reset otherwise.

RLDB Rbd, Rbs

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RLDB Rbd, Rbs 1 0 1 1 1 1 1 0 Rbs Rbd	9

Operation



Description

The contents of the source and destination byte registers are exchanged as shown in the operation. Both the source and destination are general-purpose byte registers designated by the Rbs and Rbd fields of the instruction respectively. The most significant four bits of the destination remain unchanged.

Flags

C	Z	S	P/V	DA	H
–	∗	∗	–	–	–

Z: Set to 1 if destination result is zero. Reset otherwise.
S: Set to 1 if most significant bit of destination result is 1. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

RR

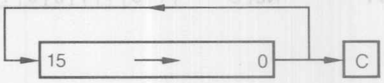
ROTATE word right

RR

RR Rd, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RR Rd, B 1 0 1 1 0 0 1 1 Rd 0 1 b 0	9 6 (one place) 7 (two places)

Operation



Description

The contents of the general-purpose word register designated by the Rd field of the instructions are rotated right. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from last bit rotated out of destination register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of destination register changed during rotation. Reset otherwise.

RRB

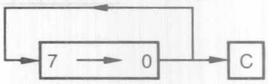
ROTATE byte right

RRB

RRB Rbd, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RRB Rbd, B 1 0 1 1 0 0 1 0 | Rbd | 0 1 b 0	6 (one place) 7 (two places)

Operation



Description

The contents of the general-purpose byte register designated by the Rbd field of the instructions are rotated right. The number of places to be rotated is specified by bit one of the instructions; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from least significant bit rotated out of destination register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of destination register changed during rotation. Reset otherwise.

RRC

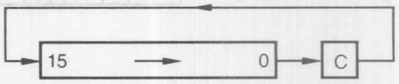
ROTATE word right through carry

RRC

RRC Rd, B

Mode	Version	Mnemonic and Form	Clocks
		RRC Rd, B	
R	NS, S	1 0 1 1 0 0 1 1 | Rd | 1 1 b 0	6 (one place) 7 (two places)

Operation



Description

The contents of the destination word register designated by the Rd field of the instruction are rotated one or two places right. The last bit rotated out of the destination word is loaded into the carry flag, while the previous contents of the carry flag are shifted into the most significant bit of the destination word. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

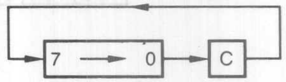
C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = *Unaffected*
1 = Set
0 = Cleared
* = Conditional − see description

C: Loaded from the last bit rotated out of destination.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of register changed during rotation. Reset otherwise.

RRCB Rbd, B

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	RRCB Rbd, B 1 0 1 1 0 0 1 0 │ Rbd │ 1 1 b 0	6 (one place) 7 (two places)	



Description

The contents of the destination byte register designated by the Rbd field of the instruction are rotated one or two places right. The last bit shifted out of the destination byte is loaded into the carry flag, while the previous contents of the carry flag are shifted into the most significant bit of the destination byte. The number of places to be rotated is specified by bit one of the instruction; zero corresponds to one place and one corresponds to two places.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into the bit field b of the instruction. The range of B is one or two, and b = B − 1. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of destination register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if sign of destination contents changes during rotation. Reset otherwise.

RRDB

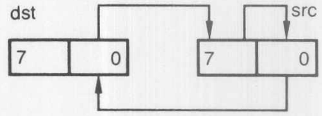
ROTATE DIGIT RIGHT, byte

RRDB

RRDB Rbd, Rbs

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	RRDB Rbd, Rbs 1 0 1 1 1 1 0 0 | Rbs | Rbd	9

Operation



Description

The contents of the source and destination byte register are exchanged as shown in the operation. Both the source and destination are general-purpose byte registers designated by the Rbs and Rbd fields of the instruction, respectively. The most significant four bits of the destination remain unchanged.

Flags

C	Z	S	P/V	DA	H
–	*	*	–	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if destination result is zero. Reset otherwise.
S: Set to 1 if most significant bit of destination result is 1. Reset otherwise.

SUBTRACT word with carry

SBC Rd, Rs

Mode	Version	Mnemonic and Form			Clocks
		SBC Rd, Rs			
R	NS, S	1 0 1 1 1 0 1 1	Rs	Rd	5

Operation

Rd<0:15>←Rd<0:15>−Rs<0:15>−C

Description

The source operand word is subtracted from the destination operand word, along with carry, to obtain the result. The subtraction is achieved by adding the two's complement of the source operand to the destination operand.

Both the source and destination are general-purpose word registers designated by the Rs and Rd fields of the instruction, respectively. The 16-bit result is loaded into the destination register, whose original contents are lost. The contents of the source are not affected.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

C: Reset to 0 on carry from most significant bit of result. Set otherwise.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if there is arithmetic overflow. Reset otherwise.

SBCB

SUBTRACT byte with carry

SBCB

SBCB Rbd, Rbs

Mode	Version	Mnemonic and Form			Clocks
		SBCB Rbd, Rbs			
R	NS, S	1 0 1 1 0 1 1 0	Rbs	Rbd	5

Operation

Rbd<0:7>←Rbd<0:7>−Rbs<0:7>−C

Description

The source operand byte is subtracted from the destination operand byte, along with carry, to obtain the result. The subtraction is achieved by adding the two's complement of the source operand to the destination operand.

Both the source and destination are general-purpose byte registers designated by the Rbs and Rbd fields of the instruction, respectively. The 8-bit result is loaded into the destination register whose original contents are lost. The contents of the source are not altered.

5

Flags

C	Z	S	P/V	DA	H
*	*	*	*	1	*

− = Unaffected
1 = Set
0 = Cleared
* = Conditional − see description

C: Reset to 0 on carry from most significant bit of result. Set otherwise.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set to 1 if result is negative. Reset otherwise.
P/V: Set to 1 if there is arithmetic overflow. Reset otherwise.
DA: Set to 1 always.
H: Reset to 0 if there is a carry from most significant bit of the lower 4 bits of the result. Set otherwise.

SC

SYSTEM CALL

SC

SC N

Mode	Version	Mnemonic and Form	Clocks
		SC N	
–	NS, S	0 1 1 1 1 1 1 1 | n	33, 39

Operation

Non-Segmented	Segmented
R15'<0:15>←R15'<0:15>−2	R15'<0:15>←R15'<0:15>−2
(R15'<0:15>)←PC<0:15>+2	(RR14'<0:22>)←PC OFFSET+2
	R15'<0:15>←R15'<0:15>−2
	(RR14'<0:22>)←PC SEGMENT
R15'<0:15>←R15'<0:15>−2	R15'<0:15>←R15'<0:15>−2
(R15'<0:15>)←FCW	(RR14'<0:22>)←FCW
R15'<0:15>←R15'<0:15>−2	R15'<0:15>←R15'<0:15>−2
(R15'<0:15>)←Identifier	(RR14'<0:22>) ←Identifier
FCW←(NPSAP<0:15>+12)	FCW ←(NPSAP<0:22>+26)
PC←(NPSAP<0:15>+14)	PC SEGMENT←(NPSAP<0:22>+28)
	PC OFFSET←(NPSAP<0:22>+30)

Description

This instruction produces a system call trap. The system call causes the program status to be pushed into the system stack and then loads the new processor status using NPSAP.

The status stored on the stack comprises the program counter return address, and the flag control word (FCW) as well as the system call instruction itself, as the Identifier.

The new program counter and FCW are obtained from the NPSAP and are loaded into the relevant CPU registers to cause the transfer of control. The 8-bit n field of the instruction is user definable, and thus allows up to 256 identifiers.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into a binary value in the n field of the instruction. The range of N is zero to 255, and n=N. Specifying an N outside the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	*	*

As specified by the new FCW.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

SDA

SHIFT word arithmetic (dynamic)

SDA

SDA Rd, Rs

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SDA Rd, Rs	15 + 3n*

1 0 1 1	0 0 1 1	Rd	1 0 1 1
▨	Rs	▨	▨

*n is the number of places shifted.

Operation

Rd<0:15>←Rd<0:15>shifted

Description

The contents of a general-purpose word register designated by the Rd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range −16 to +16.

This operation is identical to the operation SDL apart from the treatment of the most significant bit of the word, bit 15. This bit is unaltered during right shifts, and shifts into the adjacent bit, bit 14. For left shifts, the bit is treated in an identical manner to other bits of the register. Thus a signed operand has the sign preserved during right shifts.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from bit 15 shifted out of destination register (left shift) or from bit 0 shifted out of the destination register (right shift).
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if most significant bit of resultant destination register is 1. Reset otherwise.
P/V: Set to 1 if sign of destination register is changed during shift. Reset otherwise.

5

SDAB

SHIFT byte arithmetic (dynamic)

SDAB Rbd, Rs

SDAB

Mode	Version	Mnemonic and Form	Clocks
		SDAB Rbd, Rs	
R	NS, S	1 0 1 1 0 0 1 0 | Rbd | 1 0 1 1 ▨ | Rs | ▨ | ▨	15 + 3n*

1 0 1 1	0 0 1 0	Rbd	1 0 1 1
▨	Rs	▨	▨

*n is the number of places shifted.

Operation

Rbd<0:7>←Rbd<0:7>shifted

Description

The contents of a general-purpose byte register designated by the Rbd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range −8 to +8.

This operation is identical to the operation SDLB apart from the treatment of the most significant bit of the byte, bit seven. This bit is unaltered during right shifts, and shifts into the adjacent bit, bit six. For left shifts, the bit is treated in an identical manner to other bits of the register. Thus a signed operand has the sign preserved during right shifts.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

C: Loaded from bit 7 shifted out of destination register (left shift) or from bit 0 shifted out of the destination register (right shift).
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if most significant bit of resultant destination register is 1. Reset otherwise.
P/V: Set to 1 if sign of destination register is changed during shift. Reset otherwise.

SHIFT long word arithmetic (dynamic)

SDAL RRd, Rs

Mode	Version	Mnemonic and Form	Clocks	Operation
		SDAL RRd, Rs		RRd<0:31>←RRd<0:31>shifted
R	NS, S		15 + 3n*	

1	0	1	1	0	0	1	1	RRd	1 1 1 1	
▨				Rs				▨	▨	

*n is the number of places shifted.

Description

The contents of a general-purpose long word register designated by the RRd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range −32 to +32.

This operation is identical to the operation SDLL apart from the treatment of the most significant bit of the long word, bit 31. This bit is unaltered during right shifts, and shifts into the adjacent bit, bit 30. For left shifts, the bit is treated to other bits of the register. Thus a signed operand has the sign preserved during right shifts.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

C: Loaded from bit 31 shifted out of destination register (left shift) or from bit 0 shifted out of the destination register (right shift).
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if most significant bit of resultant destination register is 1. Reset otherwise.
P/V: Set to 1 if sign of destination register is changed during shift. Reset otherwise.

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	SDL Rd, Rs	15 + 3n*	Rd<0:15>←Rd<0:15>shifted

<table>
<tr><td colspan="2">1 0 1 1 0 0 1 1</td><td>Rd</td><td>0 0 1 1</td></tr>
<tr><td>▨</td><td>Rs</td><td>▨</td><td>▨</td></tr>
</table>

*n is the number of places shifted.

Description

The contents of a general-purpose word register designated by the Rd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range −16 to +16.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the destination register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Undefined.

SDLB

SHIFT byte logical (dynamic)

SDLB Rbd, Rs

SDLB

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	SDLB Rbd, Rs 1 0 1 1 0 0 1 0 | Rbd | 0 0 1 1 ▨ | Rs | ▨ | ▨	15 + 3n*	Rbd<0:7>←Rbd<0:7>(shifted)

*n is the number of places shifted.

Description

The contents of a general-purpose byte register designated by the Rbd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range −8 to +8.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	−	−

− = Unaffected
1 = Set
0 = Cleared
∗ = Conditional − see description

C: Loaded from the last bit shifted out of the destination register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Undefined.

SDLL

SHIFT long word logical (dynamic)

SDLL

SDLL RRd, Rs

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SDLL RRd, Rs	15 + 3n*

1 0 1 1	0 0 1 1	RRd	0 1 1 1
▨	Rs	▨	▨

*n is the number of places shifted.

Operation

RRd<0:31>←RRd<0:31>(shifted)

Description

The contents of a general-purpose register pair designated by the RRd field of the instruction are shifted. The magnitude and direction of the shift are determined from the contents of the general-purpose word register designated by the Rs field of the instruction. The register contains a signed two's complement integer which is used to determine the shift value. A positive number indicates a left shift, and a negative number indicates a right shift. The magnitude of the shift must be in the range −32 to +32.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of destination register pair.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Undefined.

SET

SET bit in word (static)

SET

SET dst, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SET Rd, B 1 0 1 0 0 1 0 1 | Rd | b	4
IR	NS	SET Rd↑, B 0 0 1 0 0 1 0 1 | Rd ≠ 0 | b	11
IR	S	SET RRd↑, B 0 0 1 0 0 1 0 1 | RRd ≠ 0 | b	11
DA	NS	SET LABEL, B 0 1 1 0 0 1 0 1 | 0 0 0 0 | b ADDRESS	13
DA	SSO	SET LABSSO, B 0 1 1 0 0 1 0 1 | 0 0 0 0 | b 0 | SEGMENT | OFFSET	14
DA	SLO	SET LABEL, B 0 1 1 0 0 1 0 1 | 0 0 0 0 | b 1 | SEGMENT | ▨ OFFSET	16
X	NS	SET LABEL (Rx), B 0 1 1 0 0 1 0 1 | Rx ≠ 0 | b ADDRESS	14
X	SSO	SET LABSSO (Rx), B 0 1 1 0 0 1 0 1 | Rx ≠ 0 | b 0 | SEGMENT | OFFSET	14
X	SLO	SET LABEL (Rx), B 0 1 1 0 0 1 0 1 | Rx ≠ 0 | b 1 | SEGMENT | ▨ OFFSET	17

Operation

word dst <b bit>←1

Description

The selected bit of the word destination is set to one. The remaining 15 bits are unaltered. The destination is determined by the applicable addressing mode, while the bit to be set is determined by the binary value of the b field of the instruction.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

5

SET

SET bit in word (dynamic)

SET Rd, Rs

SET

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SET Rd, Rs	10

0 0 1 0	0 1 0 1	0 0 0 0	Rs
▨	Rd	▨	▨

Operation

Rd<bit specified in Rs(0:3)>←1

Description

The selected bit of the word destination is set to one. The destination word operand is the general-purpose register designated by the Rd field of the instruction. The bit of the destination register to be set is determined by a binary decode of the least significant four bits of a general-purpose word register designated by the Rs field of the instruction. The remaining 15 bits of the destination are unaltered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SETB

SET bit in byte (static)

SETB

SET dst, B

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SET Rbd, B 1 0 1 0 0 1 0 0 | Rbd | b	4
IR	NS	SET Rd↑, B 0 0 1 0 0 1 0 0 | Rd ≠ 0 | b	11
IR	S	SET RRd↑, B 0 0 1 0 0 1 0 0 | RRd ≠ 0 | b	11
DA	NS	SET LABEL, B 0 1 1 0 0 1 0 0 | 0 0 0 0 | b ADDRESS	13
DA	SSO	SET LABSSO, B 0 1 1 0 0 1 0 0 | 0 0 0 0 | b 0 | SEGMENT | OFFSET	14
DA	SLO	SET LABEL, B 0 1 1 0 0 1 0 0 | 0 0 0 0 | b 1 | SEGMENT | ▨ OFFSET	16
X	NS	SET LABEL (Rx), B 0 1 1 0 0 1 0 0 | Rx ≠ 0 | b ADDRESS	14
X	SSO	SET LABSSO (Rx), B 0 1 1 0 0 1 0 0 | Rx ≠ 0 | b 0 | SEGMENT | OFFSET	14
X	SLO	SET LABEL (Rx), B 0 1 1 0 0 1 0 0 | Rx ≠ 0 | b 1 | SEGMENT | ▨ OFFSET	17

Operation

byte dst <b bit>←1

Description

The selected bit of the byte destination is set to one. The remaining seven bits are unaltered. The destination is determined by the applicable addressing mode, while the bit to be set to one is determined by the binary value of the b field of the instruction.

Assembler Notation

The assembler notation B is a numeric expression which is assembled into a binary value in the b field of the instruction. The range of B is zero through 15, and b = B. Specifying a B outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SETB

SET bit in byte (dynamic)

SETB

SETB Rbd, Rs

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	SETB Rbd, Rs	10	Rbd<bit specified in Rs(0:2)>←1

0,0,1,0	0,1,0,0	0,0,0,0	Rs
▨	Rbd	▨	▨

Description

The selected bit of the byte destination is set to one. The destination byte operand is the general-purpose register designated by the Rbd field of the instruction. The bit of the destination register to be set is determined by binary decode of the least significant three bits of a general-purpose word register designated by the Rs field of the instruction. The remaining seven bits of the destination are unaltered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

SETFLG

SET FLAGS

SETFLG

SET FLG LIST

Mode	Version	Mnemonic and Form	Clocks	Operation
		SETFLG LIST		FCW <C;Z;S;P/V>←1
—	NS, S	1 0 0 0 1 1 0 1 | C Z S PV | 0 0 0 1	7	(see description below)

Description

The CPU flags, C, Z, S and P/V are set or unaltered according to the bit settings in the instruction field as described in the table below.

Instruction Bit	If = 0	If = 1	Assembler Notation
7	No effect	Set C flag	CY
6	No effect	Set Z flag	ZR
5	No effect	Set S flag	SGN
4	No effect	Set P/V flag	PY or OV

Assembler Notation

The assembler notation LIST refers to a list of any or all of the following reserved words, separated by commas: CY, ZR, SGN, PY, or OV. Note that PY and OV affect the same flag.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

See description.

– = *Unaffected*
1 = Set
0 = Cleared
* = Conditional – see description

SPECIAL INPUT word to register from I/O port

SIN Rd, PORT

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
PA	NS, S	SIN Rd, PORT	12

0 0 1 1 1 0 1 1	Rd	0 1 0 1
PORT ADDRESS		

Operation

Rd <0:15>←port src<0:15>

Description

A general-purpose word destination register designated by the Rd field of the instruction is loaded from an input port. The port address is determined directly from the instruction. The original contents of the destination are lost.

The instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

SINB

SPECIAL INPUT byte to register from I/O port

SINB

SINB Rbd, PORT

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
PA	NS, S	SINB Rbd, PORT 0,0,1,1,1,0,1,0 | Rbd | 0,1,0,1 PORT ADDRESS	12	Rbd<0:7>←port src<0:7>

Description

A general-purpose byte destination register designated by the Rbd field of the instruction is loaded from an input port. The port address is determined directly from the instruction. The original contents of the destination are lost.

The instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

SIND

SPECIAL INPUT word from I/O port to memory, autodecrement

SIND

SIND dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		SIND Rd↑, Rp, Rc				
IR, PR	NS	0 0 1 1 1 0 1 1		Rp	1 0 0 1	21
		0 0 0 0	Rc	Rd	1 0 0 0	
		SIND RRd↑, Rp, Rc				
IR, PR	S	0 0 1 1 1 0 1 1		Rp	1 0 0 1	21
		0 0 0 0	Rc	RRd	1 0 0 0	

Operation

dst<0:15>←port src<0:15>
Rd<0:15>←Rd<0:15>−2
Rc<0:15>←Rc<0:15>−1

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are then decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

P/V: Set to one if the result of decrementing Rc register is zero. Reset otherwise.

SINDB SPECIAL INPUT byte from I/O port to memory, autodecrement SINDB

SINDB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		SINDB Rd↑, Rp, Rc				
IR, PR	NS	0 0 1 1 1 0 1 0		Rp	1 0 0 1	21
		0 0 0 0	Rc	Rd	1 0 0 0	
		SINDB RRd↑, Rp, Rc				
IR, PR	S	0 0 1 1 1 0 1 0		Rp	1 0 0 1	21
		0 0 0 0	Rc	RRd	1 0 0 0	

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>−1
Rc<0:15>←Rc<0:15>−1

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registers designated by Rd and Rc are then decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

P/V: Set to one if the result of decrementing Rc register is zero. Reset otherwise.

SINDR

SPECIAL INPUT word from I/O port to memory, autodecrement and repeat

SINDR

SINDR dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		SINDR Rd↑, Rp, Rc				
IR, PR	NS	0 0 1 1	1 0 1 1	Rp	1 0 0 1	11 + 10n*
		0 0 0 0	Rc	Rd	0 0 0 0	
		SINDR RRd↑, Rp, Rc				
IR, PR	S	0 0 1 1	1 0 1 1	Rp	1 0 0 1	11 + 10n*
		0 0 0 0	Rc	RRd	0 0 0 0	

*n is the number of iterations.

Operation

dst<0:15>←port src<0:15>
Rd<0:15>←Rd<0:15>−2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into the memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then decremented by two. The contents of the general-purpose register designated by Rc are then decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to one.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

SINDRB

SPECIAL INPUT byte from I/O port to memory, autodecrement and repeat

SINDRB

SINDRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	SINDRB Rd↑, Rp, Rc	11 + 10n*

0 0 1 1	1 0 1 0	Rp	1 0 0 1
0 0 0 0	Rc	Rd	0 0 0 0

Mode	Version	Mnemonic and Form	Clocks
IR, PR	S	SINDRB RRd↑, Rp, Rc	11 + 10n*

0 0 1 1	1 0 1 0	Rp	1 0 0 1
0 0 0 0	Rc	RRd	0 0 0 0

*n is the number of iterations.

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>−1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into the memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd and Rc are then decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to one.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SPECIAL INPUT word from I/O port to memory, autoincrement

SINI dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	SINI Rd↑, Rp, Rc 0,0,1,1,1,0,1,1 | Rp | 0,0,0,1 0,0,0,0 | Rc | Rd | 1,0,0,0	21	dst<0:15>←port src<0:15> Rd<0:15>←Rd<0:15>+2 Rc<0:15>←Rc<0:15>−1
IR, PR	S	SINI RRd↑, Rp, Rc 0,0,1,1,1,0,1,1 | Rp | 0,0,0,1 0,0,0,0 | Rc | RRd | 1,0,0,0	21	

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	*	–	–

P/V: Set to 1 if the result of decrementing Rc register is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

SINIB SPECIAL INPUT byte from I/O port to memory, autoincrement SINIB

SINIB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	SINIB Rd↑, Rp, Rc 0 0 1 1 1 0 1 0 Rp 0 0 0 1 0 0 0 0 Rc Rd 1 0 0 0	21
IR, PR	S	SINIB RRd↑, Rp, Rc 0 0 1 1 1 0 1 0 Rp 0 0 0 1 0 0 0 0 Rc RRd 1 0 0 0	21

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

P/V: Set to 1 if the result of decrementing Rc register is zero. Reset otherwise.

SINIR SPECIAL INPUT word from I/O port to memory, autoincrement and repeat SINIR

SINIR dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	SINIR Rd↑, Rp, Rc 0 0 1 1 1 0 1 1 | Rp | 0 0 0 1 0 0 0 0 | Rc | Rd | 0 0 0 0	11 + 10n*
IR, PR	S	SINIR RRd↑, Rp, Rc 0 0 1 1 1 0 1 1 | Rp | 0 0 0 1 0 0 0 0 | Rc | RRd | 0 0 0 0	11 + 10n*

*n is the number of iterations.

Operation

dst<0:15>← port src<0:15>
Rd<0:15>←Rd<0:15>+2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data word from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose registeɾ designated by Rd are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SINIRB — SPECIAL INPUT byte from I/O port to memory, autoincrement and repeat — SINIRB

SINIRB dst, Rp, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		SINIRB Rd↑, Rp, Rc				
IR, PR	NS	0 0 1 1	1 0 1 0	Rp	0 0 0 1	11 + 10n*
		0 0 0 0	Rc	Rd	0 0 0 0	
		SINIRB RRd↑, Rp, Rc				
IR, PR	S	0 0 1 1	1 0 1 0	Rp	0 0 0 1	11 + 10n*
		0 0 0 0	Rc	RRd	0 0 0 0	

*n is the number of iterations.

Operation

dst<0:7>←port src<0:7>
Rd<0:15>←Rd<0:15>+1
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

Data byte from the port addressed by the contents of the general-purpose register designated by the Rp field of the instruction is loaded into a memory destination. The destination is addressed by the contents of the general-purpose register designated by the Rd (or RRd) field of the instruction. The original contents of the destination are lost. The contents of the general-purpose register designated by Rd are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one. This instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose port source or destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

5

SLA

SHIFT word arithmetic left

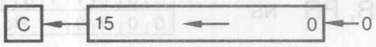
SLA Rd, N

SLA

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SLA Rd, N 1 0 1 1 1 0 0 1 1 | Rd | 1 0 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the word destination register are shifted left. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of the word register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Set to 1 if sign of register changed during shift operation. Reset otherwise.

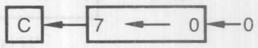
SHIFT byte arithmetic left

SLAB Rbd, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SLAB Rbd, N 1 0 1 1 0 0 1 0 | Rbd | 1 0 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the byte destination register are shifted left. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n = N. Specifying an N outside of the allowable range produces an assembler error.

5

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the byte register.
Z: Set to 1 if result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Set to 1 if sign of register changed during shift operation. Reset otherwise.

SLAL

SHIFT long word left arithmetic

SLAL

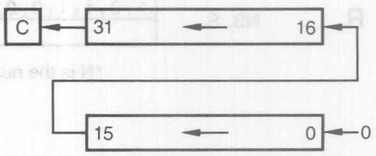
SLAL RRd, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SLAL RRd, N	13 + 3N*

1 0 1 1 0 0 1 1	RRd	1 1 0 1
n		

*N is the number of places shifted.

Operation



Description

The contents of the long word destination register are shifted left. The destination is a general-purpose long word register designated by the RRd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 32. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 32, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

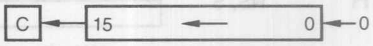
C: Loaded from the last bit shifted out of the word register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Set to 1 if sign of register changed during shift operation. Reset otherwise.

SLL

SHIFT word logical left

SLL Rd, N

SLL

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	SLL Rd, N 1 0 1 1 0 0 1 1 | Rd | 0 0 0 1 n *N is the number of places shifted.	13 + 3N*	

Description

The contents of the word destination register are shifted left. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n = N. Specifying an N outside of the allowable range produces an assembler error.

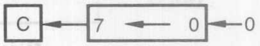
5

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the register pair.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Undefined.

Mode	Version	Mnemonic and Form	Clocks	Operation
R	NS, S	SLLB Rbd, N 1 0 1 1 0 0 1 0 | Rbd | 0 0 0 1 n *N is the number of places shifted.	13 + 3N*	

Description

The contents of the byte destination register are shifted left. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the byte register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Undefined.

SLLL

SHIFT long word logical left

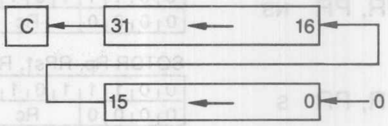
SLLL

SLLL RRd, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SLLL RRd, N 1 0 1 1 0 0 1 1 RRd 0 1 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the register pair are shifted left. The register pair is designated by the RRd field of the instruction. The magnitude of the shift is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 32. The n field is a 16-bit positive integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 32, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of the register pair.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Undefined.

SOTDR SPECIAL OUTPUT word from memory to I/O port, autodecrement and repeat SOTDR

SOTDR Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	SOTDR Rp, Rs↑, Rc 0,0,1,1,1,0,1,1 | Rs | 1,0,1,1 0,0,0,0 | Rc | Rp | 0,0,0,0	11 + 10n*
IR, PR	S	SOTDR Rp, RRs↑, Rc 0,0,1,1,1,0,1,1 | RRs | 1,0,1,1 0,0,0,0 | Rc | Rp | 0,0,0,0	11 + 10n*

*n is the number of iterations.

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>−2
Rc<0:15>←Rc<0:15>−1
Repeat until termination.

Description

A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register designated by the Rc field is decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

SOTDRB SPECIAL OUTPUT byte from memory to I/O port, autodecrement and repeat SOTDRB

SOTDRB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks	Operation
		SOTDRB Rp, Rs↑, Rc					port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>−1 Rc<0:15>←Rc<0:15>−1 Repeat until termination.
IR, PR	NS	0 0 1 1	1 0 1 0	Rs	1 0 1 1	11 + 10n*	
		0 0 0 0	Rc	Rp	0 0 0 0		
		SOTDRB Rp, RRs↑, Rc					
IR, PR	S	0 0 1 1	1 0 1 0	RRs	1 0 1 1	11 + 10n*	
		0 0 0 0	Rc	Rp	0 0 0 0		

*n is the number of iterations.

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by the Rc field is decremented by one. The instruction is terminated when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

SOTIR **SPECIAL OUTPUT** word from memory to I/O port, autoincrement and repeat SOTIR

SOTIR Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		SOTIR Rp, Rs↑, Rc				
IR, PR	NS	0011	1011	Rs	0011	11 + 10n*
		0000	Rc	Rp	0000	
		SOTIR Rp, RRs↑, Rc				
IR, PR	S	0011	1011	RRs	0011	11 + 10n*
		0000	Rc	RRp	0001	

*n is the number of iterations.

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>+2
Rc<0:15>←Rc<0:15>−1

Description

A data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by the Rc are decremented by one. The instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

SOTIRB

SPECIAL OUTPUT byte from memory to I/O port, autoincrement and repeat

SOTIRB

SOTIRB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	SOTIRB Rp, Rs↑, Rc 0 0 1 1 | 1 0 1 0 | Rs | 0 0 1 1 0 0 0 0 | Rc | Rp | 0 0 0 0	11 + 10n*
IR, PR	S	SOTIRB Rp, RRs↑, Rc 0 0 1 1 | 1 0 1 0 | RRs | 0 0 1 1 0 0 0 0 | Rc | RRp | 0 0 0 0	11 + 10n*

*n is the number of iterations.

Operation

port dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

A data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by the Rc are decremented by one. The instruction terminates when the result of this decrementation reaches zero.

This instruction is interruptible.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	1	–	–

P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SOUT

SPECIAL OUTPUT word from register to I/O port

SOUT

SOUT PORT, Rs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
PA	NS, S	SOUT PORT, Rs 0 0 1 1 1 0 1 1 | Rs | 0 1 1 1 PORT ADDRESS	12	port dst<0:15>←Rbs<0:15>

Description

The contents of the general-purpose word source register designated by the Rs field of the instruction are loaded into an output port. The port address is determined directly from the instruction. The source contents are unaltered.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST_0-ST_3.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
= Conditional – see description

SOUTB

SPECIAL OUTPUT byte from register to I/O port

SOUTB

SOUTB PORT, Rbs

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
		SOUTB PORT, Rbs		port dst<0:7>←Rbs<0:7>
PA	NS, S		12	

0,0,1,1,1,0,1,0	Rbs	0,1,1,1
PORT ADDRESS		

Description

The contents of the general-purpose byte source register designated by the Rbs field of the instruction are loaded into an output port. The port address is determined directly from the instruction. The source contents are unaltered.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SOUTD

SPECIAL OUTPUT word from memory to I/O port, autodecrement

SOUTD

SOUTD Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks
IR, PR	NS	SOUTD Rp, Rs↑, Rc 0 0 1 1 1 0 1 1 | Rs | 1 0 1 1 0 0 0 0 | Rc | Rp | 1 0 0 0	21
IR, PR	S	SOUTD Rp, RRs↑, Rc 0 0 1 1 1 0 1 1 | RRs | 1 0 1 1 0 0 0 0 | Rc | Rp | 1 0 0 0	21

Operation

port dst<0:15>←src<0:15>
Rs<0:15>←Rs<0:15>−2
Rc<0:15>←Rc<0:15>−1

Description

Data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rd field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	✱	–	–

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

SOUTDB

SPECIAL OUTPUT byte from memory to I/O port, autodecrement

SOUTDB

SOUTDB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	SOUTDB Rp, Rs↑, Rc 0 0 1 1 1 0 1 0 Rs 1 0 1 1 0 0 0 0 Rc Rp 1 0 0 0	21	port dst<0:7>←src<0:7> Rs<0:15>←Rs<0:15>−1 Rc<0:15>←Rc<0:15>−1
IR, PR	S	SOUTDB Rp, RRs↑, Rc 0 0 1 1 1 0 1 0 RRs 1 0 1 1 0 0 0 0 Rc Rp 1 0 0 0	21	

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rd field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then decremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

5

SOUTI

SPECIAL OUTPUT word from memory to I/O port, autoincrement

SOUTI

SOUTI Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form	Clocks	Operation
IR, PR	NS	SOUTI Rp, Rs↑, Rc 0 0 1 1 1 0 1 1 · Rs · 0 0 1 1 0 0 0 0 · Rc · Rp · 1 0 0 0	21	port dst<0:15>←src<0:15> Rs<0:15>←Rs<0:15>+2 Rc<0:15>←Rc<0:15>−1
IR, PR	S	SOUTI Rp, RRs↑, Rc 0 0 1 1 1 0 1 1 · RRs · 0 0 1 1 0 0 0 0 · Rc · Rp · 1 0 0 0	21	

Description

Data word in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by two. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

Flags

C	Z	S	P/V	DA	H
–	–	–	∗	–	–

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

SOUTIB

SPECIAL OUTPUT byte from memory to I/O port, autoincrement

SOUTIB

SOUTIB Rp, src, Rc

This is a SYSTEM instruction.

Mode	Version	Mnemonic and Form				Clocks
		SOUTIB Rp, Rs↑, Rc				
IR, PR	NS	0 0 1 1 1 0 1 0		Rs	0 0 1 1	21
		0 0 0 0	Rc	Rp	1 0 0 0	
		SOUTIB Rp, RRs↑, Rc				
IR, PR	S	0 0 1 1 1 0 1 0		RRs	0 0 1 1	21
		0 0 0 0	Rc	Rp	1 0 0 0	

Operation

port dst<0:7>←src<0:7>
Rs<0:15>←Rs<0:15>+1
Rc<0:15>←Rc<0:15>−1

Description

Data byte in memory, addressed by the contents of the general-purpose register designated by the Rs (or RRs) field of the instruction, is loaded into the destination port. The destination is addressed by the contents of the general-purpose register designated by the Rp field of the instruction. The source contents are unaltered. The contents of the general-purpose register designated by Rs are then incremented by one. The contents of the general-purpose register designated by Rc are decremented by one.

This instruction is similar in operation to the corresponding standard I/O instruction. The only difference is the value on the CPU status lines ST_0-ST_3.

This instruction uses both indirect register memory addressing and port register port addressing modes.

R0 can be designated as the general-purpose source or port destination register.

5

Flags

C	Z	S	P/V	DA	H
–	–	–	*	–	–

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

SRA

SHIFT word arithmetic right

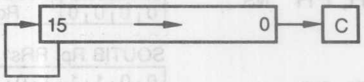
SRA Rd, N

SRA

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SRA Rd, N 1 0 1 1 0 0 1 1 | Rd | 1 0 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the word destination register are shifted right. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit negative integer in two's complement notation.

This operation is identical to the operation SRL apart from the treatment of the most significant bit of the word, bit 15. This bit is unaltered during the shift operation, and shifts into the adjacent bit, bit 14. Thus a signed operand has the sign preserved during the shifting operation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	0	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the word register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Reset.

SRAB

SHIFT byte arithmetic right

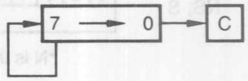
SRAB

SRAB Rbd, N

Mode	Version	Mnemonic and Form	Clocks
		SRAB Rbd, N	
R	NS, S	1 0 1 1 0 0 1 0 | Rbd | 1 0 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the byte destination register are shifted right. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit negative integer in two's complement notation.

This operation is identical to the operation SRLB apart from the treatment of the most significant bit of the byte, bit seven. This bit is unaltered during the shift operation, and shifts into the adjacent bit, bit six. Thus a signed operand has its sign preserved during the shifting operation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	0	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of the byte register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Reset.

SRAL

SHIFT long word right arithmetic

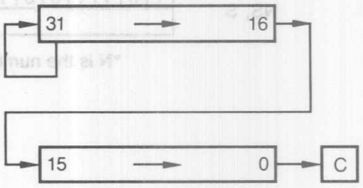
SRAL

SRAL RRd, N

Mode	Version	Mnemonic and Form	Clocks
		SRAL RRd, N	
R	NS, S	1 0 1 1 0 0 1 1 | RRd | 1 1 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the long word destination register are shifted right. The destination is a general-purpose long word register designated by the RRd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 32. The n field is a 16-bit negative integer in two's complement notation.

This operation is identical to the operation SRLL apart from the treatment of the most significant bit of the long word, bit 31. This bit is unaltered during the shift operation, and shifts into the adjacent bit, bit 30. Thus a signed operand has its sign preserved during the shifting operation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 32, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	0	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the register pair.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Reset.

SHIFT word logical right

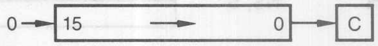
SRL Rd, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SRL Rd, N	13 + 3N*

1	0	1	1	1	0	0	1	1	Rd	0	0	0	1
n													

*N is the number of places shifted.

Operation



Description

The contents of the word destination register are shifted right. The destination is a general-purpose word register designated by the Rd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 16. The n field is a 16-bit negative integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 16, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Loaded from the last bit shifted out of the word register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Undefined.

SRLB

SHIFT byte logical right

SRLB

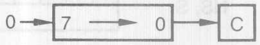
SRLB Rbd, N

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SRLB Rbd, N	13 + 3N*

1	0	1	1	0	0	1	0	Rbd	0	0	0	1
n												

*N is the number of places shifted.

Operation



Description

The contents of the byte destination register are shifted right. The destination is a general-purpose byte register designated by the Rbd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to eight. The n field is a 16-bit negative integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to eight, and n = N. Specifying an N outside of the allowable range produces an assembler error.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of the byte register.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Undefined.

SRLL

SHIFT long word logical right (static)

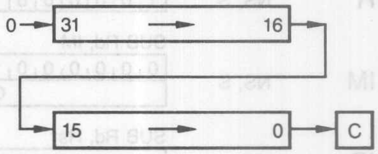
SRLL

SRLL RRd, N

Mode	Version	Mnemonic and Form	Clocks
		SRLL RRd, N	
R	NS, S	1 0 1 1 1 0 0 1 1 | RRd | 0 1 0 1 n	13 + 3N*

*N is the number of places shifted.

Operation



Description

The contents of the long word destination register are shifted right. The destination is a general-purpose register pair designated by the RRd field of the instruction. The number of places to be shifted is determined from the value of the n field of the instruction. The magnitude of the shift must be in the range zero to 32. The n field is a 16-bit negative integer in two's complement notation.

Assembler Notation

The assembler notation N is a numeric expression which is assembled into the bit field n of the instruction. The range of N is zero to 32, and n = N. Specifying an N outside of the allowable range produces an assembler error.

The number n is a negative two's complement number.

Flags

C	Z	S	P/V	DA	H
∗	∗	∗	∗	–	–

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

C: Loaded from the last bit shifted out of the register pair.
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set if the most significant bit of the resultant destination is 1. Reset otherwise.
P/V: Undefined.

SUB

SUBTRACT word from register

SUB

SUB Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SUB Rd, Rs 1 0 0 0 0 0 1 1 | Rs | Rd	4
IM	NS, S	SUB Rd, IM 0 0 0 0 0 0 1 1 | 0 0 0 0 | Rd OPERAND	7
IR	NS	SUB Rd, Rs↑ 0 0 0 0 0 0 1 1 | Rs ≠ 0 | Rd	7
IR	S	SUB Rd, RRs↑ 0 0 0 0 0 0 1 1 | RRs ≠ 0 | Rd	7
DA	NS	SUB Rd, LABEL 0 1 0 0 0 0 1 1 | 0 0 0 0 | Rd ADDRESS	9
DA	SSO	SUB Rd, LABSSO 0 1 0 0 0 0 1 1 | 0 0 0 0 | Rd 0 | SEGMENT | OFFSET	10
DA	SLO	SUB Rd, LABEL 0 1 0 0 0 0 1 1 | 0 0 0 0 | Rd 1 | SEGMENT | ▨ OFFSET	12
X	NS	SUB Rd, LABEL (Rx) 0 1 0 0 0 0 1 1 | Rx ≠ 0 | Rd ADDRESS	10
X	SSO	SUB Rd, LABSSO (Rx) 0 1 0 0 0 0 1 1 | Rx ≠ 0 | Rd 0 | SEGMENT | OFFSET	10
X	SLO	SUB Rd, LABEL (Rx) 0 1 0 0 0 0 1 1 | Rx ≠ 0 | Rd 1 | SEGMENT | ▨ OFFSET	13

Operation

Rd<0:15>←Rd<0:15>−src<0:15>

Description

The source word operand contents are subtracted from the contents of the general-purpose word register designated by the Rd field of the instruction. The result is loaded into the destination. The source operand is obtained using the applicable addressing mode. The original contents of the destination register are lost while those of the source operand are unaltered.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

C: Reset on carry from the most significant bit of result. Set to 1 otherwise (i.e., borrow).
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

SUBB

SUBTRACT byte from register

SUBB

SUBB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SUBB Rbd, Rbs 1 0 0 0 0 0 1 0 · Rbs · Rbd	4
IM	NS, S	SUBB Rbd, IMb 0 0 0 0 0 0 1 0 · 0 0 0 0 · Rbd 7 OPERAND 0 · 7 OPERAND 0	7
IR	NS	SUBB Rbd, Rs↑ 0 0 0 0 0 0 1 0 · Rs ≠ 0 · Rbd	7
IR	S	SUBB Rbd, RRs↑ 0 0 0 0 0 0 1 0 · RRs ≠ 0 · Rbd	7
DA	NS	SUBB Rbd, LABEL 0 1 0 0 0 0 1 0 · 0 0 0 0 · Rbd ADDRESS	9
DA	SSO	SUBB Rbd, LABSSO 0 1 0 0 0 0 1 0 · 0 0 0 0 · Rbd 0 · SEGMENT · OFFSET	10
DA	SLO	SUBB Rbd, LABEL 0 1 0 0 0 0 1 0 · 0 0 0 0 · Rbd 1 · SEGMENT · ▨ OFFSET	12
X	NS	SUBB Rbd, LABEL (Rx) 0 1 0 0 0 0 1 0 · Rx ≠ 0 · Rbd ADDRESS	10
X	SSO	SUBB Rbd, LABSSO (Rx) 0 1 0 0 0 0 1 0 · Rx ≠ 0 · Rbd 0 · SEGMENT · OFFSET	10
X	SLO	SUBB Rbd, LABEL (Rx) 0 1 0 0 0 0 1 0 · Rx ≠ 0 · Rbd 1 · SEGMENT · ▨ OFFSET	13

Operation

Rbd<0:7>←Rbd<0:7>−src<0:7>

Description

The source byte operand contents are subtracted from the contents of the general-purpose byte register designated by the Rbd field of the instruction. The result is loaded into the destination. The source operand is obtained using the applicable addressing mode. The original contents of the destination register are lost and those of the source operand are unaltered.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	1	*

− = Unaffected
1 = Set
0 = Cleared
* = Conditional − see description

C: Reset on carry from the most significant bit of result. Set to 1 otherwise (i.e., borrow).
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.
DA: Set to 1 always.
H: Reset on carry from most significant bit of lower 4 bits of result. Set otherwise (i.e., borrow).

5

SUBL

SUBTRACT long word from register

SUBL

SUBL RRd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	SUBL RRd, RRs | 1,0,0,1,0,0,1,0 | RRs | RRd |	8
IM	NS, S	SUBL RRd, IMℓ | 0,0,0,1,0,0,1,0 | 0,0,0,0 | RRd | | 31 OPERAND 16 | | 15 OPERAND 0 |	14
IR	NS	SUBL RRd, Rs↑ | 0,0,0,1,0,0,1,0 | Rs ≠ 0 | RRd |	14
IR	S	SUBL RRd, RRs↑ | 0,0,0,1,0,0,1,0 | RRs ≠ 0 | RRd |	14
DA	NS	SUBL RRd, LABEL | 0,1,0,1,0,0,1,0 | 0,0,0,0 | RRd | | ADDRESS |	15
DA	SSO	SUBL RRd, LABSSO | 0,1,0,1,0,0,1,0 | 0,0,0,0 | RRd | | 0 | SEGMENT | OFFSET |	16
DA	SLO	SUBL RRd, LABEL | 0,1,0,1,0,0,1,0 | 0,0,0,0 | RRd | | 1 | SEGMENT | ▨ | | OFFSET |	18
X	NS	SUBL RRd, LABEL (Rx) | 0,1,0,1,0,0,1,0 | Rx ≠ 0 | RRd | | ADDRESS |	16
X	SSO	SUBL RRd, LABSSO (Rx) | 0,1,0,1,0,0,1,0 | Rx ≠ 0 | RRd | | 0 | SEGMENT | OFFSET |	16
X	SLO	SUBL RRd, LABEL (Rx) | 0,1,0,1,0,0,1,0 | Rx ≠ 0 | RRd | | 1 | SEGMENT | ▨ | | OFFSET |	19

Operation

RRd<0:31>←RRd<0:31>−src<0:31>

Description

The source long word operand contents are subtracted from the contents of the general-purpose register pair designated by the RRd field of the instruction. The result is loaded into the destination. The source operand is obtained using the applicable addressing mode. The original contents of the destination register are lost while those of the source operand are unaltered.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	–	–

– = *Unaffected*
1 = Set
0 = Cleared
* = Conditional – see description

C: Reset on carry from the most significant bit of result. Set to 1 otherwise (i.e., borrow).
Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Set to 1 on arithmetic overflow. Reset otherwise.

TCC

TEST condition codes and set a bit in word

TCC CC, Rd

TCC

Mode	Version	Mnemonic and Form	Clocks
		TCC CC, Rd	
R	NS, S	1 0 1 0 1 1 1 1 | Rd | CC	5

Operation

Rd<bit 0>←1 if condition is met.

Description

The contents of the flags are compared with those specified by the CC field of the instruction. If the comparison is successful, the least significant bit of the destination word register Rd is set to one. Otherwise this bit is unaffected. Remaining bits of the destination are not altered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

TCCB — TEST condition codes and set a bit in byte — TCCB

TCCB CC, Rbd

Mode	Version	Mnemonic and Form			Clocks
		TCCB CC, Rbd			
R	NS, S	1 0 1 0 1 1 1 0	Rbd	CC	5

Operation

Rbd<bit 0>←1 if condition is met.

Description

The contents of the flags are compared with those specified by the CC field of the instruction. If the comparison is successful, the least significant bit of the destination byte register Rbd is set to one. Otherwise this bit is unaffected. Remaining bits of the destination are not altered.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

TEST

TEST word

TEST

TEST dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	TEST Rd 1 0 0 0 1 1 0 1 | Rd | 0 1 0 0	7
IR	NS	TEST Rd↑ 0 0 0 0 1 1 0 1 | Rd | 0 1 0 0	8
IR	S	TEST RRd↑ 0 0 0 0 1 1 0 1 | RRd | 0 1 0 0	8
DA	NS	TEST LABEL 0 1 0 0 1 1 0 1 | 0 0 0 0 | 0 1 0 0 ADDRESS	11
DA	SSO	TEST LABSSO 0 1 0 0 1 1 0 1 | 0 0 0 0 | 0 1 0 0 0 | SEGMENT | OFFSET	12
DA	SLO	TEST LABEL 0 1 0 0 1 1 0 1 | 0 0 0 0 | 0 1 0 0 1 | SEGMENT | ▨ OFFSET	14
X	NS	TEST LABEL (Rx) 0 1 0 0 1 1 0 1 | Rx ≠ 0 | 0 1 0 0 ADDRESS	12
X	SSO	TEST LABSSO (Rx) 0 1 0 0 1 1 0 1 | Rx ≠ 0 | 0 1 0 0 0 | SEGMENT | OFFSET	12
X	SLO	TEST LABEL (Rx) 0 1 0 0 1 1 0 1 | Rx ≠ 0 | 0 1 0 0 1 | SEGMENT | ▨ OFFSET	15

Operation

dst<0:15>←dst<0:15> V 0

Description

The contents of the destination word operand are tested to set the appropriate flags. Testing is done by performing a logical OR operation between destination word and zero. The destination is determined by the applicable addressing mode and the contents of the destination are not altered.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	*	*	–	–	–

Z: Set to 1 if the result is zero. Reset otherwise.

S: Set to 1 if the result is negative. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

TESTB

TEST byte

TESTB

TESTB dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	TESTB Rbd 1 0 0 0 1 1 0 0 │ Rbd │ 0 1 0 0	7
IR	NS	TESTB Rd↑ 0 0 0 0 1 1 0 0 │ Rd │ 0 1 0 0	8
IR	S	TESTB RRd↑ 0 0 0 0 1 1 0 0 │ RRd │ 0 1 0 0	8
DA	NS	TESTB LABEL 0 1 0 0 1 1 0 0 │ 0 0 0 0 │ 0 1 0 0 ADDRESS	11
DA	SSO	TESTB LABSSO 0 1 0 0 1 1 0 0 │ 0 0 0 0 │ 0 1 0 0 0 │ SEGMENT │ OFFSET	12
DA	SLO	TESTB LABEL 0 1 0 0 1 1 0 0 │ 0 0 0 0 │ 0 1 0 0 1 │ SEGMENT │ ▨ OFFSET	14
X	NS	TESTB LABEL (Rx) 0 1 0 0 1 1 0 0 │ Rx ≠ 0 │ 0 1 0 0 ADDRESS	12
X	SSO	TESTB LABSSO (Rx) 0 1 0 0 1 1 0 0 │ Rx ≠ 0 │ 0 1 0 0 0 │ SEGMENT │ OFFSET	12
X	SLO	TESTB LABEL (Rx) 0 1 0 0 1 1 0 0 │ Rx ≠ 0 │ 0 1 0 0 1 │ SEGMENT │ ▨ OFFSET	15

Operation

dst<0:7>←dst<0:7> V 0

Description

The contents of the destination byte operand are tested to set the appropriate flags. Testing is done by performing a logical OR operation between destination byte and zero. The destination is determined by the applicable addressing mode and the contents of the destination are not altered.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	*	*	*	–	–

Z: Set to 1 if the operand is zero. Reset otherwise.
S: Set to 1 if the operand is negative. Reset otherwise.
P/V: Set to 1 if parity of operand is even. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

TESTL

TEST long word

TESTL dst

TESTL

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	TESTL RRd 1 0 0 1 1 1 0 0 | RRd | 1 0 0 0	13
IR	NS	TESTL Rd↑ 0 0 0 1 1 1 0 0 | Rd | 1 0 0 0	13
IR	S	TESTL RRd↑ 0 0 0 1 1 1 0 0 | RRd | 1 0 0 0	13
DA	NS	TESTL LABEL 0 1 0 1 1 1 0 0 | 0 0 0 0 | 1 0 0 0 ADDRESS	16
DA	SSO	TESTL LABSSO 0 1 0 1 1 1 0 0 | 0 0 0 0 | 1 0 0 0 0 | SEGMENT | OFFSET	17
DA	SLO	TESTL LABEL 0 1 0 1 1 1 0 0 | 0 0 0 0 | 1 0 0 0 1 | SEGMENT | ▨ OFFSET	19
X	NS	TESTL LABEL (Rx) 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 1 0 0 0 ADDRESS	17
X	SSO	TESTL LABSSO (Rx) 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 1 0 0 0 0 | SEGMENT | OFFSET	17
X	SLO	TESTL LABEL (Rx) 0 1 0 1 1 1 0 0 | Rx ≠ 0 | 1 0 0 0 1 | SEGMENT | ▨ OFFSET	20

Operation

dst<0:31>←dst<0:31> V 0

Description

The contents of the long word destination are tested to set the appropriate flags. Testing is done by performing a logical OR operation between destination and zero. The destination is determined by the applicable addressing mode and the contents of the destination are not altered.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	*	*	–		

Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

TRDB

TRANSLATE byte, autodecrement

TRDB

TRDB dst, src, Rc

Mode	Version	Mnemonic and Form	Clocks	Operation
				(see description below)
IR	NS	TRDB Rd↑, Rs↑, Rc 1 0 1 1 1 0 0 0 · Rd · 1 0 0 0 0 0 0 0 · Rc · Rs · 0 0 0 0	25	
IR	S	TRDB RRd↑ RRs↑, Rc 1 0 1 1 1 0 0 0 · RRd · 1 0 0 0 0 0 0 0 · Rc · RRs · 0 0 0 0	25	

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRDRB. The contents of register RH1 are undefined following TRDB.

R0 can be designated as the general-purpose source designation register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Undefined.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

TRDRB

TRANSLATE byte, autodecrement and repeat

TRDRB

TRDRB dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks
		TRDRB Rd↑, Rs↑, Rc				
IR	NS	1 0 1 1 1 0 0 0		Rd	1 1 0 0	11 + 14n*
		0 0 0 0	Rc	Rs	0 0 0 0	
		TRDRB RRd↑, RRs↑, Rc				
IR	S	1 0 1 1 1 0 0 0		RRd	1 1 0 0	11 + 14n*
		0 0 0 0	Rc	Rs	0 0 0 0	

*n is the number of iterations.

Operation

(see description below)

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction repeats until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration. The contents of register RH1 are undefined following TRDRB.

R0 can be designated as the general-purpose source or destination register.

5

Flags

C	Z	S	P/V	DA	H
–	*	–	1	–	–

Z: Undefined.
P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

TRANSLATE byte, autoincrement

TRIB, dst, src, Rc

Mode	Version	Mnemonic and Form	Clocks	Operation
				(see description below)
IR	NS	TRIB Rd↑, Rs↑, Rc	25	
IR	S	TRIB RRd↑, RRs↑, Rc	25	

TRIB Rd↑, Rs↑, Rc

1,0,1,1,1,0,0,0		Rd	0,0,0,0
0,0,0,0	Rc	Rs	0,0,0,0

TRIB RRd↑, RRs↑, Rc

1,0,1,1,1,0,0,0		RRd	0,0,0,0
0,0,0,0	Rc.	RRs	0,0,0,0

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRIRB. The contents of register RH1 are undefined following TRIB.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	∗	–	∗	–	–

Z: Undefined.
P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

TRANSLATE byte string, autoincrement and repeat

TRIRB dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks
		TRIRB Rd↑, Rs↑, Rc				
IR	NS	1,0,1,1,1,0,0,0		Rd	0,1,0,0	11 + 14n*
		0,0,0,0	Rc	Rs	0,0,0,0	
		TRIRB RRd↑, RRs↑, Rc				
IR	S	1,0,1,1,1,0,0,0		RRd	0,1,0,0	11 + 14n*
		0,0,0,0	Rc	RRs	0,0,0,0	

*n is the number of iterations.

Operation

(see description below)

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the address specified by the Rd register.

The address specified by the Rd register is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated. This completes one iteration.

This instruction repeats until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration. The contents of register RH1 are undefined following TRIRB.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	*	–	1	–	–

Z: Undefined.
P/V: Set to 1.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

TRTDB

TRANSLATE AND TEST byte, autodecrement

TRTDB

TRTDB dst, src, Rc

Mode	Version	Mnemonic and Form			Clocks	Operation
		TRTDB Rd↑, Rs↑, Rc				(see description below)
IR	NS	1 0 1 1 1 0 0 0	Rd	1 0 1 0	25	
		0 0 0 0 Rc	Rs	0 0 0 0		
		TRTDB RRd↑, RRs↑, Rc				
IR	S	1 0 1 1 1 0 0 0	RRd	1 0 1 0	25	
		0 0 0 0 Rc	RRs	0 0 0 0		

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRTDRB.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	D A	H
–	*	–	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if the translated byte is zero. Reset otherwise.
P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

TRTDRB

TRANSLATE AND TEST byte, autodecrement and repeat

TRTDRB

TRTDRB dst, src, Rc

Mode	Version	Mnemonic and Form				Clocks	Operation
		TRTDRB Rd↑, Rs↑, Rc					(see description below)
IR	NS	1 0 1 1	1 0 0 0	Rd	1 1 1 0	11 + 14n*	
		0 0 0 0	Rc	Rs	1 1 1 0		
		TRTDRB RRd↑, RRs↑, Rc					
IR	S	1 0 1 1	1 0 0 0	RRd	1 1 1 0	11 + 14n*	
		0 0 0 0	Rc	RRs	1 1 1 0		

*n is the number of iterations.

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd register is decremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

The instruction repeats until the value loaded into the RH1 register is non-zero or until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if the translated byte is zero. Reset otherwise.

P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

5

TRTIB

TRANSLATE AND TEST byte, autoincrement

TRTIB

TRTIB dst, src, Rc

Mode	**Version**	**Mnemonic and Form**				**Clocks**
		TRTIB Rd↑, Rs↑, Rc				
IR	NS	1,0,1,1,1,0,0,0		Rd	0,0,1,0	25
		0,0,0,0	Rc	Rs	0,0,0,0	
		TRTIB RRd↑, RRs↑, Rc				
IR	S	1,0,1,1,1,0,0,0		RRd	0,0,1,0	25
		0,0,0,0	Rc	RRs	0,0,0,0	

Operation

(see description below)

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address, and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd register is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

This instruction terminates after one iteration. It is a special case of the instruction TRTIRB.

R0 can be designated as the general-purpose source or destination register.

Flags

C	Z	S	P/V	DA	H
–	*	–	*	–	–

Z: Set to 1 if the translated byte is zero. Reset otherwise.
P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = *Unaffected*
1 = Set
0 = Cleared
* = Conditional – see description

TRTIRB

TRANSLATE AND TEST byte string, autoincrement and repeat

TRTIRB

TRTIRB dst, src, Rc

Mode	Version	Mnemonic and Form	Clocks
		TRTIRB Rd↑, Rs↑, Rc	
IR	NS	1 0 1 1 1 0 0 0 | Rd | 0 1 1 0 0 0 0 0 | Rc | Rs | 1 1 1 0	11 + 14n*
		TRTIRB RRd↑, RRs↑, Rc	
IR	S	1 0 1 1 1 0 0 0 | RRd | 0 1 1 0 0 0 0 0 | Rc | RRs | 1 1 1 0	11 + 14n*

*n is the number of iterations.

Operation

(see description below)

Description

The general-purpose register (register pair in AmZ8001) designated by the Rs (or RRs) field of the instruction contains the starting address of a byte table.

The general-purpose register (register pair in AmZ8001) designated by the Rd (or RRd) field of the instruction contains the address of a byte string to be translated and tested.

The general-purpose register designated by the Rc field of the instruction contains the length (in bytes) of the string remaining to be translated and tested.

A byte is read from the address specified by the Rd register. This byte is used as a table index and is added to the starting address of the table. The translated byte is read from this address and is loaded into the general-purpose byte register RH1 for testing.

The address specified by the Rd field is incremented by one to point to the next byte of the string. The contents of the register designated by the Rc field of the instruction are decremented by one, to indicate the remaining length of the string to be translated and tested. This completes one iteration.

This instruction repeats until the value loaded into the RH1 register is non-zero, or until the contents of the Rc register reach zero, indicating that the string has been exhausted. This instruction is interruptible at the end of each iteration.

R0 can be designated as the general-purpose source or destination register.

5

Flags

C	Z	S	P/V	DA	H
–	∗	–	∗	–	–

Z: Set to 1 if the table entry is zero. Reset otherwise.
P/V: Set to 1 if the result of decrementing Rc is zero. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

TSET

TEST word and set

TSET

TSET dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	TSET Rd 1,0,0,0,1,1,0,1 · Rd · 0,1,1,0	7
IR	NS	TSET Rd↑ 0,0,0,0,1,1,0,1 · Rd · 0,1,1,0	11
IR	S	TSET RRd↑ 0,0,0,0,1,1,0,1 · RRd · 0,1,1,0	11
DA	NS	TSET LABEL 0,1,0,0,1,1,0,1 · 0,0,0,0 · 0,1,1,0 ADDRESS	14
DA	SSO	TSET LABSSO 0,1,0,0,1,1,0,1 · 0,0,0,0 · 0,1,1,0 0 SEGMENT · OFFSET	15
DA	SLO	TSET LABEL 0,1,0,0,1,1,0,1 · 0,0,0,0 · 0,1,1,0 1 SEGMENT · ▨ OFFSET	17
X	NS	TSET LABEL (Rx) 0,1,0,0,1,1,0,1 · Rx ≠ 0 · 0,1,1,0 ADDRESS	15
X	SSO	TSET LABSSO (Rx) 0,1,0,0,1,1,0,1 · Rx ≠ 0 · 0,1,1,0 0 SEGMENT · OFFSET	15
X	SLO	TSET LABEL (Rx) 0,1,0,0,1,1,0,1 · Rx ≠ 0 · 0,1,1,0 1 SEGMENT · ▨ OFFSET	18

Operation

If dst<0:15>←is negative, then S flag←1; otherwise S flag←0.

dst<0:15>←FFFF

Description

The most significant (sign) bit of the destination word is loaded into the S flag. The contents of the destination are then set to all ones. The destination is determined by the applicable addressing mode.

In the IR mode, R0 (or RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	∗	–	–	–

S: Set to 1 if the most significant bit of the destination is one. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

TSETB

TEST byte and set

TSETB

TSETB dst

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	TSETB Rbd 1 0 0 0 1 1 0 0 Rbd 0 1 1 0	7
IR	NS	TSETB Rd↑ 0 0 0 0 1 1 0 0 Rd 0 1 1 0	11
IR	S	TSETB RRd↑ 0 0 0 0 1 1 0 0 RRd 0 1 1 0	11
DA	NS	TSETB LABEL 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 ADDRESS	14
DA	SSO	TSETB LABSSO 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 SEGMENT OFFSET	15
DA	SLO	TSETB LABEL 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 SEGMENT ▨ OFFSET	17
X	NS	TSETB LABEL (Rx) 0 1 0 0 1 1 0 0 Rx ≠ 0 0 1 1 0 ADDRESS	15
X	SSO	TSETB LABSSO (Rx) 0 1 0 0 1 1 0 0 Rx ≠ 0 0 1 1 0 0 SEGMENT OFFSET	15
X	SLO	TSETB LABEL (Rx) 0 1 0 0 1 1 0 0 Rx ≠ 0 0 1 1 0 1 SEGMENT ▨ OFFSET	18

Operation

If dst<0:7> is negative,
then S flag←1;
otherwise S flag←0.

dst<0:7>←FF

Description

The most significant (sign) bit of the destination byte is loaded into the S flag. The contents of the destination are then set to all ones. The destination is determined by the applicable addressing mode.

In the IR mode, R0 (and RR0) can be designated as the general-purpose destination register.

Flags

C	Z	S	P/V	DA	H
–	–	*	–	–	–

S: Set to 1 if the most significant bit of the destination is 1. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

5

XOR

EXCLUSIVE OR word with register

XOR

XOR Rd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	XOR Rd, Rs 1 0 0 0 1 0 0 1 Rs Rd	4
IM	NS, S	XOR Rd, IM 0 0 0 0 1 0 0 1 0 0 0 0 Rd OPERAND	7
IR	NS	XOR Rd, Rs↑ 0 0 0 0 1 0 0 1 Rs ≠ 0 Rd	7
IR	S	XOR Rd, RRs↑ 0 0 0 0 1 0 0 1 RRs ≠ 0 Rd	7
DA	NS	XOR Rd, LABEL 0 1 0 0 1 0 0 1 0 0 0 0 Rd ADDRESS	9
DA	SSO	XOR Rd, LABSSO 0 1 0 0 1 0 0 1 0 0 0 0 Rd 0 SEGMENT OFFSET	10
DA	SLO	XOR Rd, LABEL 0 1 0 0 1 0 0 1 0 0 0 0 Rd 1 SEGMENT ▨ OFFSET	12
X	NS	XOR Rd, LABEL (Rx) 0 1 0 0 1 0 0 1 Rx ≠ 0 Rd ADDRESS	10
X	SSO	XOR Rd, LABSSO (Rx) 0 1 0 0 1 0 0 1 Rx ≠ 0 Rd 0 SEGMENT OFFSET	10
X	SLO	XOR Rd, LABEL (Rx) 0 1 0 0 1 0 0 1 Rx ≠ 0 Rd 1 SEGMENT ▨ OFFSET	13

Operation

Rd<0:15>←src<0:15> ⊕ Rd<0:15>

Description

A logical EXCLUSIVE OR operation is performed between corresponding bits of the source and destination words. The source operand is obtained by the appropriate addressing mode, and the destination operand is always a general-purpose word register designated by the Rd field of the instruction. The 16-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.

Flags

C	Z	S	P/V	DA	H
–	∗	∗	–	–	–

Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

XORB

EXCLUSIVE OR byte with register

XORB

XORB Rbd, src

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	XORB Rbd, Rbs 1 0 0 0 1 0 0 0 | Rbs | Rbd	4
IM	NS, S	XORB Rbd, IMb 0 0 0 0 1 0 0 0 | 0 0 0 0 | Rbd 7 OPERAND 0 | 7 OPERAND 0	7
IR	NS	XORB Rbd, Rs↑ 0 0 0 0 1 0 0 0 | Rs ≠ 0 | Rbd	7
IR	S	XORB Rbd, RRs↑ 0 0 0 0 1 0 0 0 | RRs ≠ 0 | Rbd	7
DA	NS	XORB Rbd, LABEL 0 1 0 0 1 0 0 0 | 0 0 0 0 | Rbd ADDRESS	9
DA	SSO	XORB Rbd, LABSSO 0 1 0 0 1 0 0 0 | 0 0 0 0 | Rbd 0 | SEGMENT | OFFSET	10
DA	SLO	XORB Rbd, LABEL 0 1 0 0 1 0 0 0 | 0 0 0 0 | Rbd 1 | SEGMENT | ▨ OFFSET	12
X	NS	XORB Rbd, LABEL (Rx) 0 1 0 0 1 0 0 0 | Rx ≠ 0 | Rbd ADDRESS	10
X	SSO	XORB Rbd, LABSSO (Rx) 0 1 0 0 1 0 0 0 | Rx ≠ 0 | Rbd 0 | SEGMENT | OFFSET	10
X	SLO	XORB Rbd, LABEL (Rx) 0 1 0 0 1 0 0 0 | Rx ≠ 0 | Rbd 1 | SEGMENT | ▨ OFFSET	13

Operation

Rbd<0:7>←src<0:7> ⊕ Rbd<0:7>

Description

A logical EXCLUSIVE OR operation is performed between corresponding bits of the source and destination bytes. The souce operand is obtained by the appropriate addressing mode, and the destination operand is always a general-purpose byte register designated by the Rbd field of the instruction. The 8-bit result is loaded into the destination, whose original contents are lost. The contents of the source are not altered.

5

Flags

C	Z	S	P/V	DA	H
–	*	*	*	–	–

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Z: Set to 1 if the result is zero. Reset otherwise.
S: Set to 1 if the result is negative. Reset otherwise.
P/V: Set to 1 if parity of result is even. Reset otherwise.

5.8 EXTENDED PROCESSING INSTRUCTIONS

The following pages include "templates" for the AmZ8001 and AmZ8002 extended processing instructions. These templates correspond to Extended Processing Architecture (EPA) instructions, which combine Extended Processing Unit (EPU) operations with possible transfers between memory and an EPU, between CPU registers and EPU registers, and between the flag byte of the CPU's FCW and the EPU.

Each of these templates is described on the following pages. The description assumes that the EPE control bit in the CPU's FCW has been set to 1. In addition, the description is from the point of view of the CPU – that is, only CPU activities are described; the operation of the EPU is implied, but the full specification of the instruction depends upon the implementation of the EPU and is beyond the scope of this manual.

Fields ignored by the CPU are shaded in the diagrams of the templates. The 2-bit field in bit positions 0 and 1 of the first word of each template would normally be used as an identification field for selecting one of up to four EPUs in a multiple EPU system configuration. Other shaded fields would typically contain opcodes for instructing an EPU as to the operation it is to perform in addition to the data transfer specified by the template.

LOAD Memory from EPU

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
IR	NS	0 0 0 0 1 1 1 1 · Rd ≠ 0 · 1 1 · ▨ ▨ · n − 1	11 + 3n
IR	S	0 0 0 0 1 1 1 1 · RRd ≠ 0 · 1 1 · ▨ ▨ · n − 1	11 + 3n
DA	NS	0 1 0 0 1 1 1 1 · 0 0 0 0 · 1 1 · ▨ ▨ · n − 1 ADDRESS	15 + 3n
DA	SSO	0 1 0 0 1 1 1 1 · 0 0 0 0 · 1 1 · ▨ ▨ · n − 1 0 · SEGMENT · OFFSET	15 + 3n
DA	SLO	0 1 0 0 1 1 1 1 · 0 0 0 0 · 1 1 · ▨ ▨ · n − 1 1 · SEGMENT · ▨ OFFSET	18 + 3n
X	NS	0 1 0 0 1 1 1 1 · Rx ≠ 0 · 1 1 · ▨ ▨ · n − 1 ADDRESS	14 + 3n
X	SSO	0 1 0 0 1 1 1 1 · Rx ≠ 0 · 1 1 · ▨ ▨ · n − 1 1 · SEGMENT · OFFSET	15 + 3n
X	SLO	0 1 0 0 1 1 1 1 · Rx ≠ 0 · 1 1 · ▨ ▨ · n − 1 1 · SEGMENT · ▨ OFFSET	17 + 3n

Operation

memory ← EPU

Description

The CPU performs the indicated address calculation and generates n EPU memory write transactions. The n words are supplied by an EPU and are stored in n consecutive memory locations starting with the effective address.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

LOAD EPU from memory

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
IR	NS	0,0,0,0,1,1,1,1 | Rs ≠ 0 | 0,1 | ▨ ▨ | n − 1	11 + 3n
IR	S	0,0,0,0,1,1,1,1 | RRs ≠ 0 | 0,1 | ▨ ▨ | n − 1	11 + 3n
DA	NS	0,1,0,0,1,1,1,1 | 0,0,0,0 | 0,1 | ▨ ▨ | n − 1 ADDRESS	15 + 3n
DA	SSO	0,1,0,0,1,1,1,1 | 0,0,0,0 | 0,1 | ▨ ▨ | n − 1 0 | SEGMENT | OFFSET	15 + 3n
DA	SLO	0,1,0,0,1,1,1,1 | 0,0,0,0 | 0,1 | ▨ ▨ | n − 1 1 | SEGMENT | ▨ OFFSET	18 + 3n
X	NS	0,1,0,0,1,1,1,1 | Rx ≠ 0 | 0,1 | ▨ ▨ | n − 1 ADDRESS	14 + 3n
X	SSO	0,1,0,0,1,1,1,1 | Rx ≠ 0 | 0,1 | ▨ ▨ | n − 1 0 | SEGMENT | OFFSET	15 + 3n
X	SLO	0,1,0,0,1,1,1,1 | Rx ≠ 0 | 0,1 | ▨ ▨ | n − 1 1 | SEGMENT | ▨ OFFSET	17 + 3n

Operation

EPU ← memory

Description

The CPU performs the indicated address calculation and generates n EPU memory read transactions. The n consecutive words are fetched from the memory locations starting with the effective address. The data is read by an EPU and operated upon according to the extended processing instruction encoded into the shaded fields.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

LOAD EPU from CPU

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	1 0 0 0 1 1 1 1 0 ▨ 1 0 ▨ ▨ src ▨ n − 1	11 + 3n

EPU← CPU registers

Description

The contents of n words are transferred to an EPU from consecutive CPU registers starting with register src. CPU registers are transferred consecutively, with register zero following register 15.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

– = Unaffected
1 = Set
0 = Cleared
✱ = Conditional – see description

Flags are not affected.

LOAD EPU from FCW

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	1 0 0 0 | 1 1 1 0 | ▨ | 1 0 | ▨ ▨ | 0 0 0 0 | ▨ | 0 0 0 0	14

Operation

EPU ← flags

Description

The flags in the CPU's FCW are transferred to an EPU on address data lines AD_0-AD_7.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

Internal EPU operation

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	1 0 0 0 1 1 1 0 ▨ 0 1 ▨ ▨	14

Operation

internal EPU operation

Description

The CPU treats this as a No Op. It is typically used to initiate an internal EPU operation.

Flags

C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = Unaffected
1 = Set
0 = Cleared
∗ = Conditional – see description

5

LOAD CPU from EPU

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S	1 0 0 0 1 1 1 1 | 1 | ▨ | 0 0 | ▨ ▨ | dst | ▨ | n − 1	11 + 3n

Operation

CPU ← EPU registers

Description

The contents of n words are transferred from an EPU to consecutive CPU registers starting with register dst. CPU registers are transferred consecutively, with register zero following register 15.

Flags

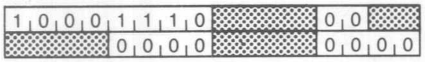
C	Z	S	P/V	DA	H
–	–	–	–	–	–

Flags are not affected.

– = *Unaffected*
1 = Set
0 = Cleared
∗ = Conditional – see description

LOAD FCW from EPU

This is an EXTENDED instruction.

Mode	Version	Mnemonic and Form	Clocks
R	NS, S		14

Operation

flags ← EPU

Description

The flags in the CPU's FCW are loaded with information from an EPU on address lines AD_0-AD_7.

The contents of CPU register zero are undefined after the execution of this instruction.

Flags

C	Z	S	P/V	DA	H
*	*	*	*	*	*

See description.

– = Unaffected
1 = Set
0 = Cleared
* = Conditional – see description

INTRODUCTION 1

CPU ARCHITECTURE 2

ADDRESSING AND DATA ORGANIZATION 3

INSTRUCTION SET ORGANIZATION 4

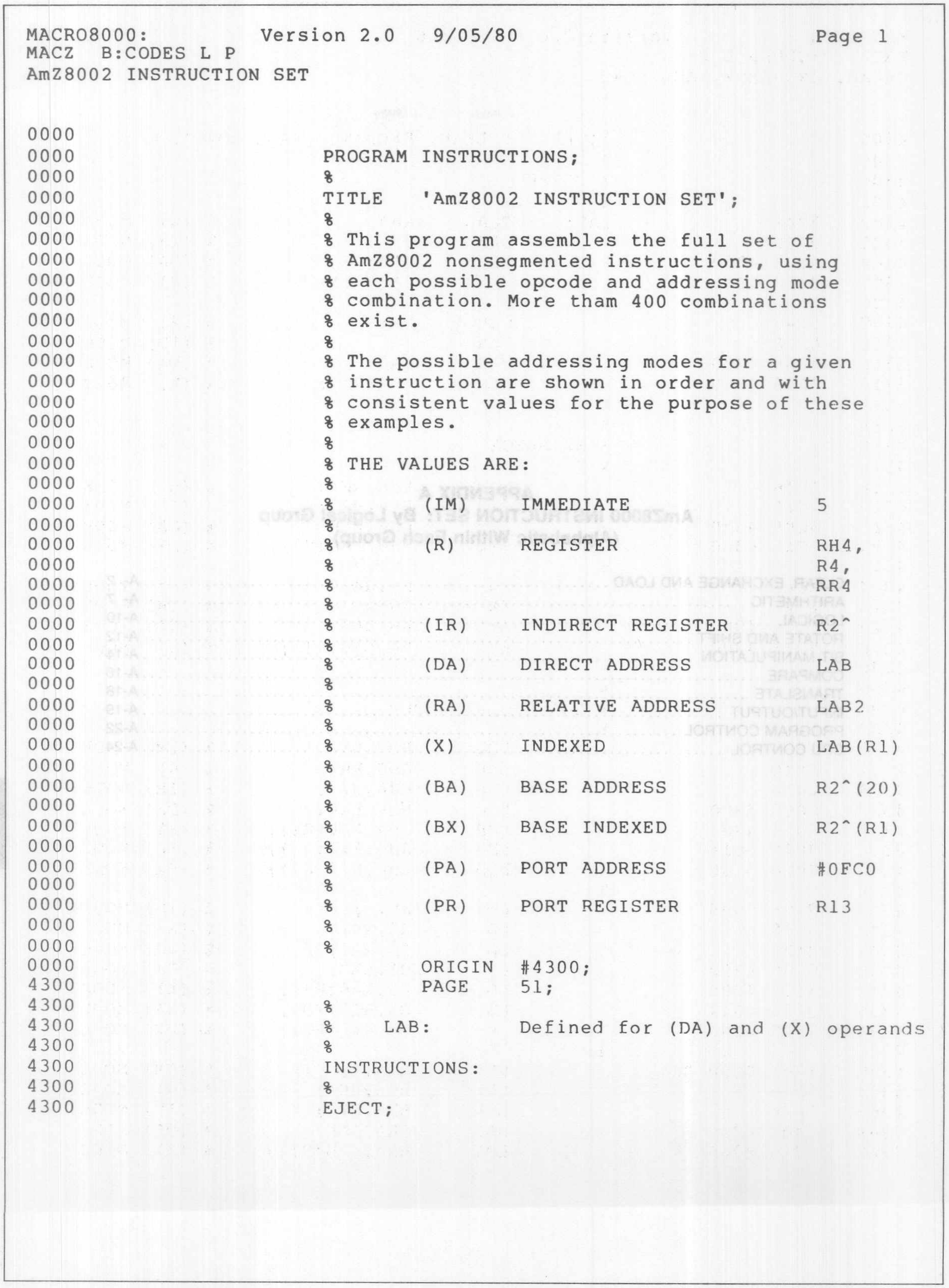
INSTRUCTION SET DETAILS 5

APPENDICES A

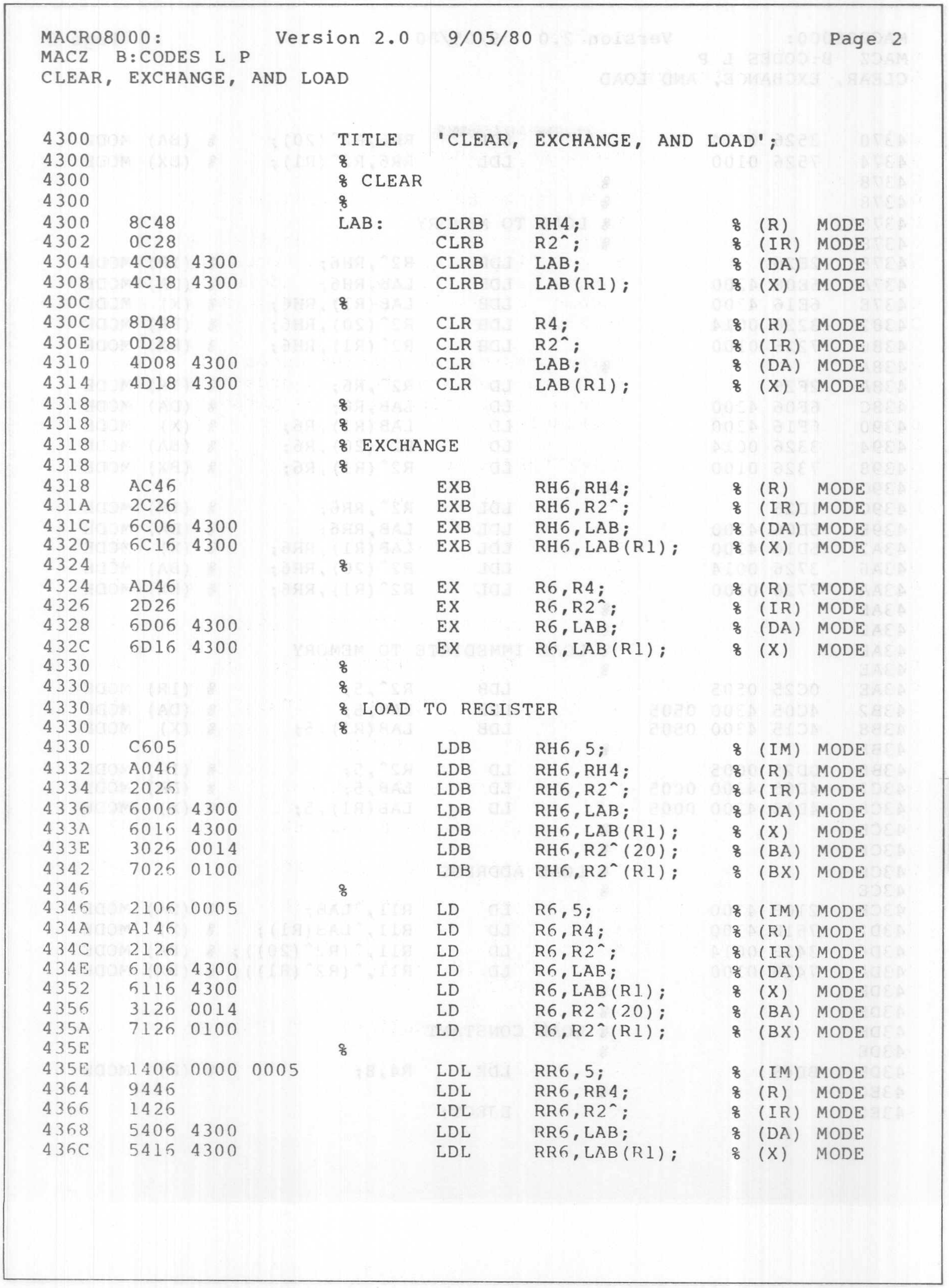
APPENDICES

A

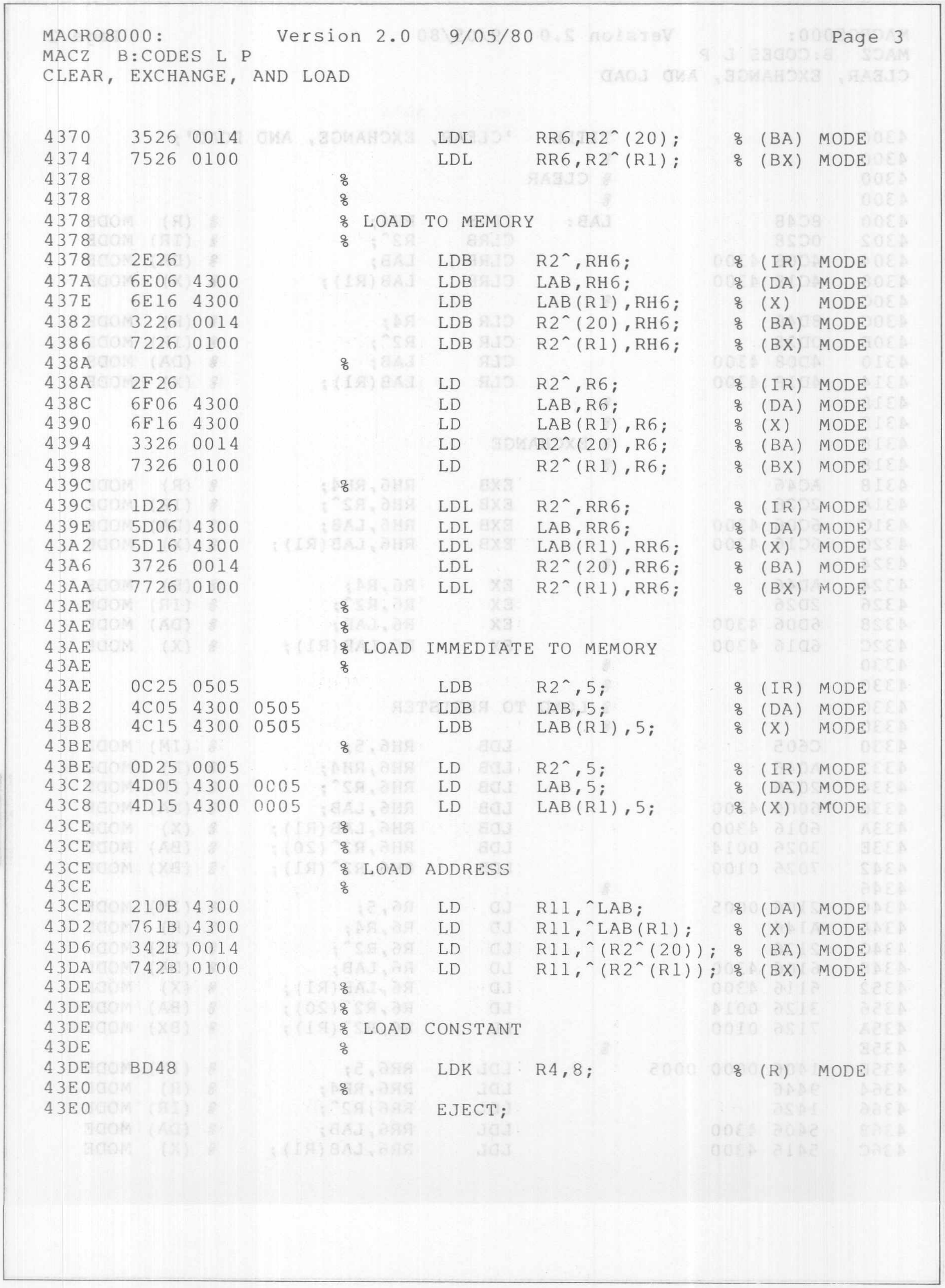
APPENDIX A
AmZ8000 INSTRUCTION SET: By Logical Group
(Alphabetic Within Each Group)



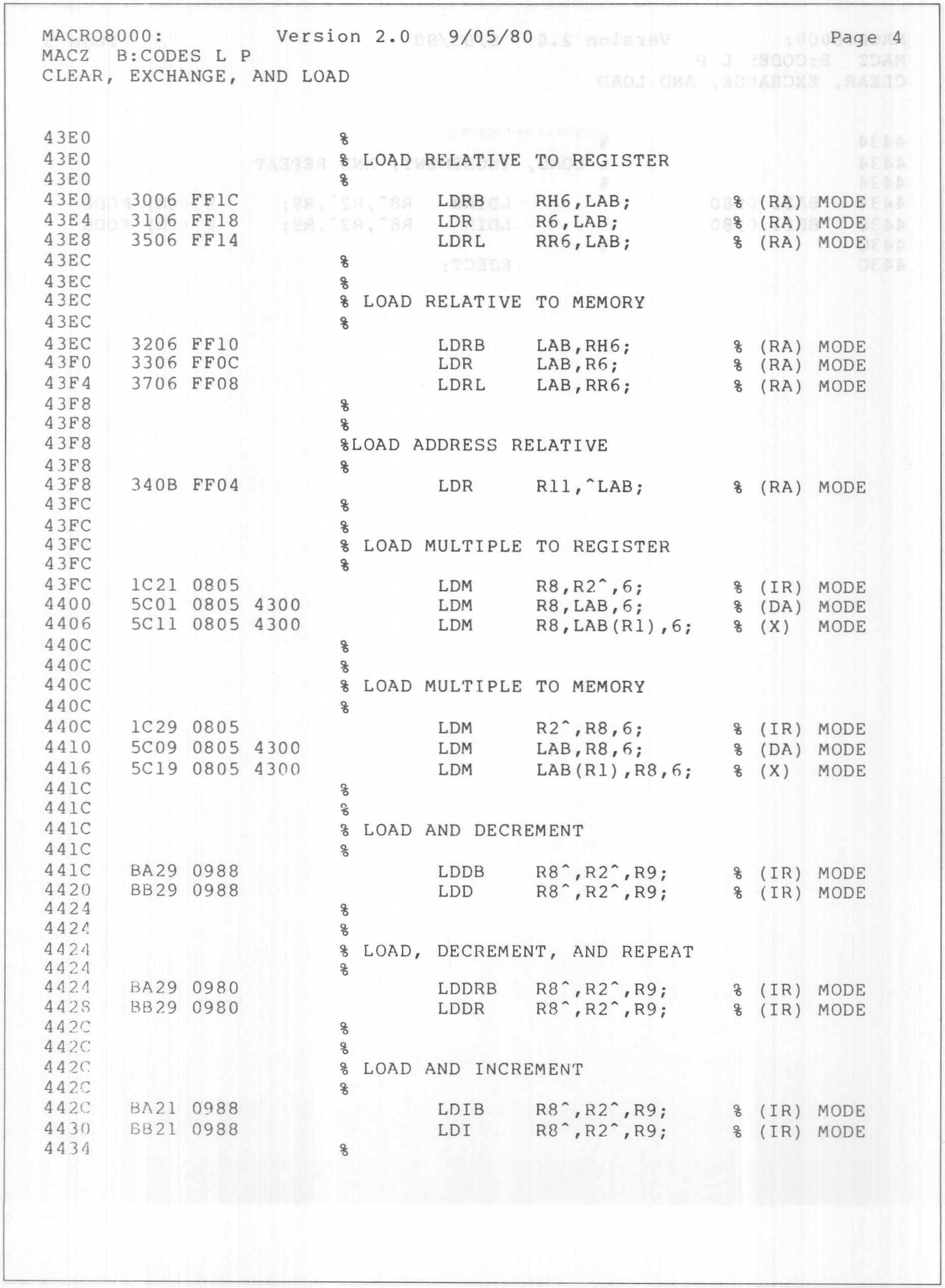
```
MACRO8000:          Version 2.0   9/05/80                      Page 1
MACZ  B:CODES L P
AmZ8002 INSTRUCTION SET

0000
0000                    PROGRAM INSTRUCTIONS;
0000                    %
0000                    TITLE   'AmZ8002 INSTRUCTION SET';
0000                    %
0000                    % This program assembles the full set of
0000                    % AmZ8002 nonsegmented instructions, using
0000                    % each possible opcode and addressing mode
0000                    % combination. More tham 400 combinations
0000                    % exist.
0000                    %
0000                    % The possible addressing modes for a given
0000                    % instruction are shown in order and with
0000                    % consistent values for the purpose of these
0000                    % examples.
0000                    %
0000                    % THE VALUES ARE:
0000                    %
0000                    %      (IM)    IMMEDIATE               5
0000                    %
0000                    %      (R)     REGISTER                RH4,
0000                    %                                      R4,
0000                    %                                      RR4
0000                    %
0000                    %      (IR)    INDIRECT REGISTER       R2^
0000                    %
0000                    %      (DA)    DIRECT ADDRESS          LAB
0000                    %
0000                    %      (RA)    RELATIVE ADDRESS        LAB2
0000                    %
0000                    %      (X)     INDEXED                 LAB(R1)
0000                    %
0000                    %      (BA)    BASE ADDRESS            R2^(20)
0000                    %
0000                    %      (BX)    BASE INDEXED            R2^(R1)
0000                    %
0000                    %      (PA)    PORT ADDRESS            #0FC0
0000                    %
0000                    %      (PR)    PORT REGISTER           R13
0000                    %
0000                    %
0000                           ORIGIN  #4300;
4300                           PAGE    51;
4300                    %
4300                    %   LAB:       Defined for (DA) and (X) operands
4300                    %
4300                    INSTRUCTIONS:
4300                    %
4300                    EJECT;
```



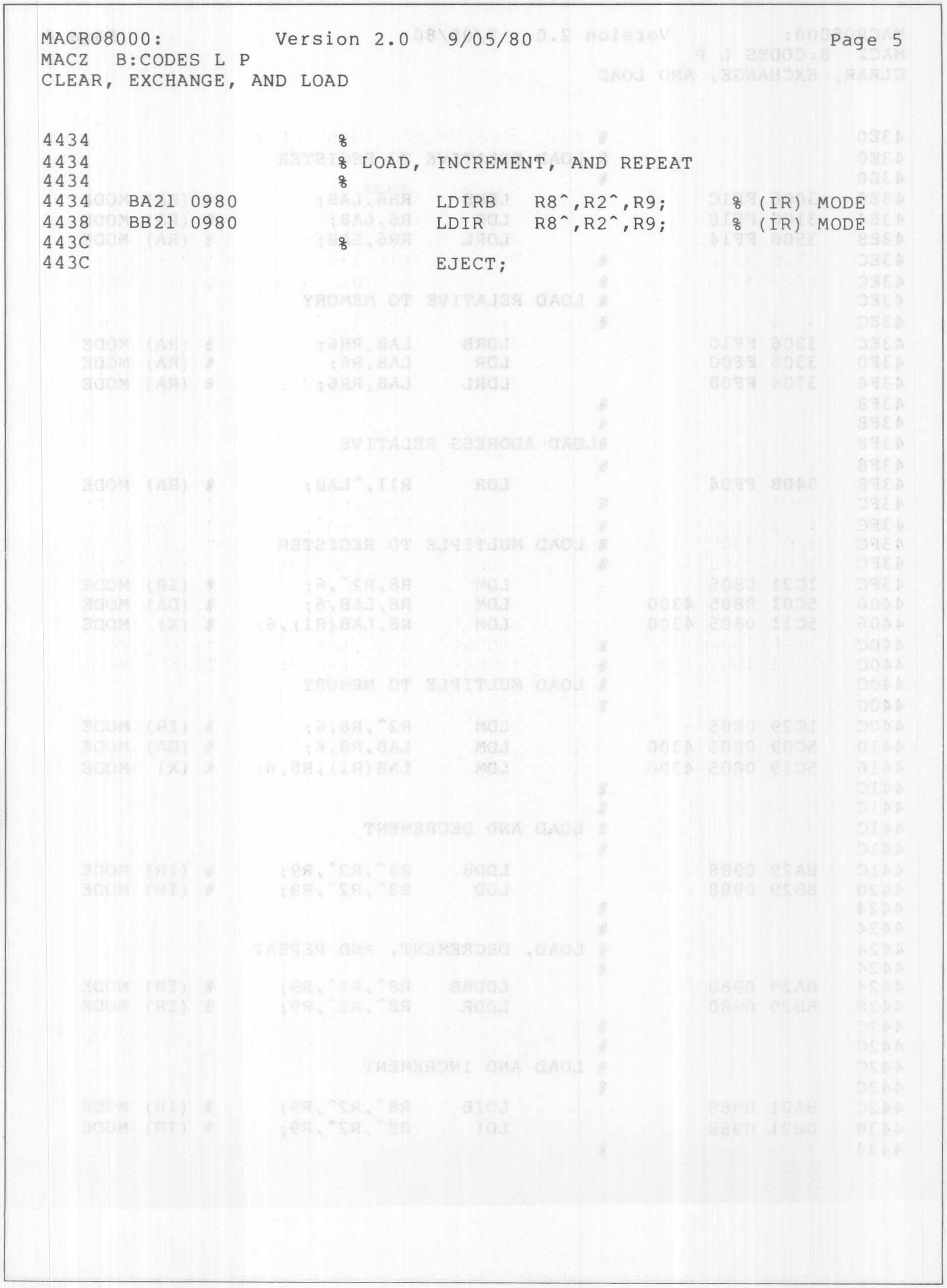
```
MACRO8000:         Version 2.0     9/05/80                        Page 2
MACZ  B:CODES L P
CLEAR, EXCHANGE, AND LOAD

4300                    TITLE   'CLEAR, EXCHANGE, AND LOAD';
4300                    %
4300                    % CLEAR
4300                    %
4300   8C48             LAB:    CLRB    RH4;             % (R)  MODE
4302   0C28                     CLRB    R2^;             % (IR) MODE
4304   4C08 4300                CLRB    LAB;             % (DA) MODE
4308   4C18 4300                CLRB    LAB(R1);         % (X)  MODE
430C                    %
430C   8D48                     CLR     R4;              % (R)  MODE
430E   0D28                     CLR     R2^;             % (IR) MODE
4310   4D08 4300                CLR     LAB;             % (DA) MODE
4314   4D18 4300                CLR     LAB(R1);         % (X)  MODE
4318                    %
4318                    %
4318                    % EXCHANGE
4318                    %
4318   AC46                     EXB     RH6,RH4;         % (R)  MODE
431A   2C26                     EXB     RH6,R2^;         % (IR) MODE
431C   6C06 4300                EXB     RH6,LAB;         % (DA) MODE
4320   6C16 4300                EXB     RH6,LAB(R1);     % (X)  MODE
4324                    %
4324   AD46                     EX      R6,R4;           % (R)  MODE
4326   2D26                     EX      R6,R2^;          % (IR) MODE
4328   6D06 4300                EX      R6,LAB;          % (DA) MODE
432C   6D16 4300                EX      R6,LAB(R1);      % (X)  MODE
4330                    %
4330                    %
4330                    % LOAD TO REGISTER
4330                    %
4330   C605                     LDB     RH6,5;           % (IM) MODE
4332   A046                     LDB     RH6,RH4;         % (R)  MODE
4334   2026                     LDB     RH6,R2^;         % (IR) MODE
4336   6006 4300                LDB     RH6,LAB;         % (DA) MODE
433A   6016 4300                LDB     RH6,LAB(R1);     % (X)  MODE
433E   3026 0014                LDB     RH6,R2^(20);     % (BA) MODE
4342   7026 0100                LDB     RH6,R2^(R1);     % (BX) MODE
4346                    %
4346   2106 0005                LD      R6,5;            % (IM) MODE
434A   A146                     LD      R6,R4;           % (R)  MODE
434C   2126                     LD      R6,R2^;          % (IR) MODE
434E   6106 4300                LD      R6,LAB;          % (DA) MODE
4352   6116 4300                LD      R6,LAB(R1);      % (X)  MODE
4356   3126 0014                LD      R6,R2^(20);      % (BA) MODE
435A   7126 0100                LD      R6,R2^(R1);      % (BX) MODE
435E                    %
435E   1406 0000 0005           LDL     RR6,5;           % (IM) MODE
4364   9446                     LDL     RR6,RR4;         % (R)  MODE
4366   1426                     LDL     RR6,R2^;         % (IR) MODE
4368   5406 4300                LDL     RR6,LAB;         % (DA) MODE
436C   5416 4300                LDL     RR6,LAB(R1);     % (X)  MODE
```



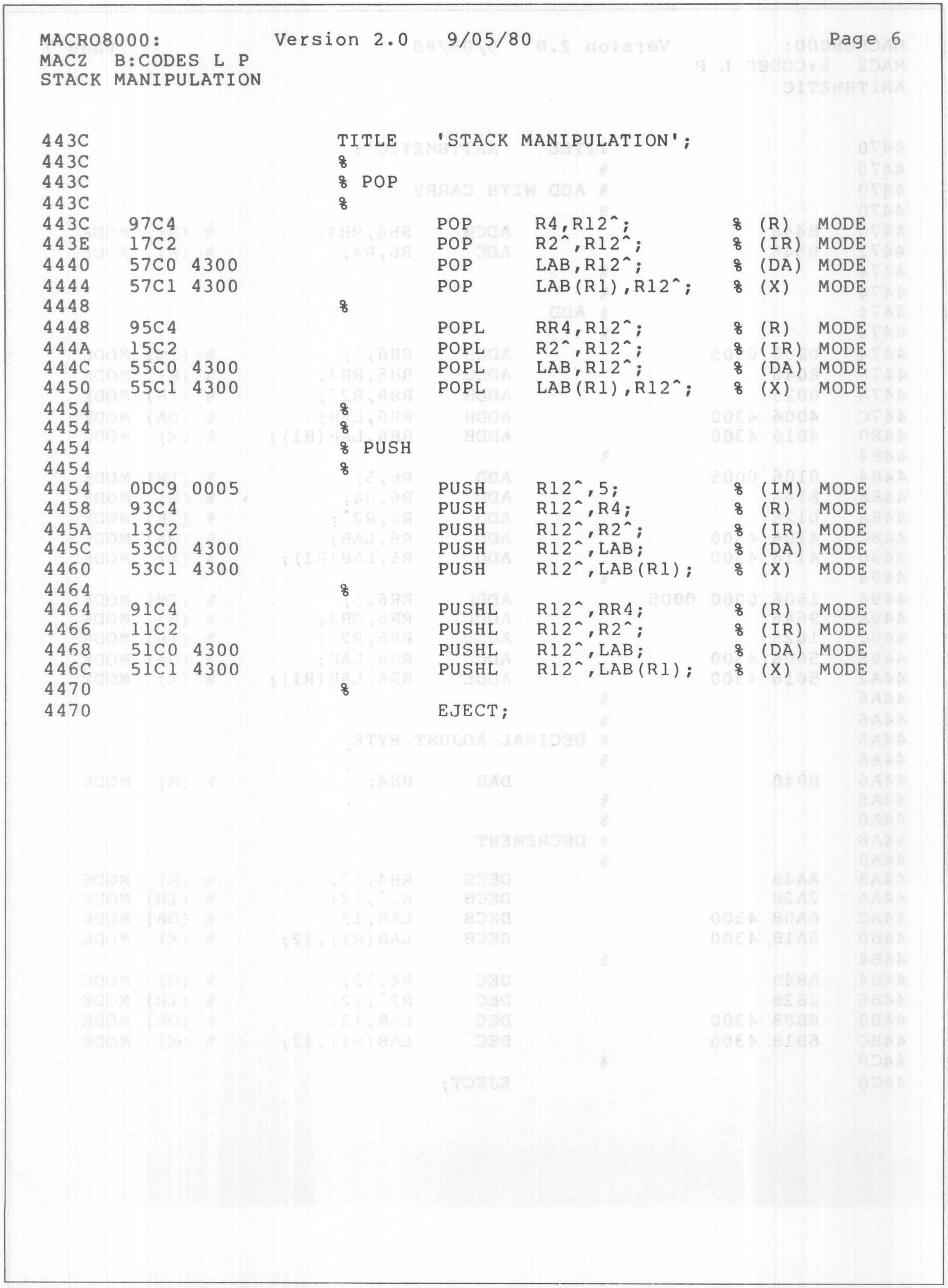
```
MACRO8000:          Version 2.0   9/05/80                         Page 3
MACZ  B:CODES L P
CLEAR, EXCHANGE, AND LOAD

4370    3526 0014                  LDL      RR6,R2^(20);        % (BA) MODE
4374    7526 0100                  LDL      RR6,R2^(R1);        % (BX) MODE
4378                   %
4378                   %
4378                   % LOAD TO MEMORY
4378                   %
4378    2E26                       LDB      R2^,RH6;            % (IR) MODE
437A    6E06 4300                  LDB      LAB,RH6;            % (DA) MODE
437E    6E16 4300                  LDB      LAB(R1),RH6;        % (X)  MODE
4382    3226 0014                  LDB      R2^(20),RH6;        % (BA) MODE
4386    7226 0100                  LDB      R2^(R1),RH6;        % (BX) MODE
438A                   %
438A    2F26                       LD       R2^,R6;             % (IR) MODE
438C    6F06 4300                  LD       LAB,R6;             % (DA) MODE
4390    6F16 4300                  LD       LAB(R1),R6;         % (X)  MODE
4394    3326 0014                  LD       R2^(20),R6;         % (BA) MODE
4398    7326 0100                  LD       R2^(R1),R6;         % (BX) MODE
439C                   %
439C    1D26                       LDL      R2^,RR6;            % (IR) MODE
439E    5D06 4300                  LDL      LAB,RR6;            % (DA) MODE
43A2    5D16 4300                  LDL      LAB(R1),RR6;        % (X)  MODE
43A6    3726 0014                  LDL      R2^(20),RR6;        % (BA) MODE
43AA    7726 0100                  LDL      R2^(R1),RR6;        % (BX) MODE
43AE                   %
43AE                   %
43AE                   % LOAD IMMEDIATE TO MEMORY
43AE                   %
43AE    0C25 0505                  LDB      R2^,5;              % (IR) MODE
43B2    4C05 4300 0505             LDB      LAB,5;              % (DA) MODE
43B8    4C15 4300 0505             LDB      LAB(R1),5;          % (X)  MODE
43BE                   %
43BE    0D25 0005                  LD       R2^,5;              % (IR) MODE
43C2    4D05 4300 0005             LD       LAB,5;              % (DA) MODE
43C8    4D15 4300 0005             LD       LAB(R1),5;          % (X)  MODE
43CE                   %
43CE                   %
43CE                   % LOAD ADDRESS
43CE                   %
43CE    210B 4300                  LD       R11,^LAB;           % (DA) MODE
43D2    761B 4300                  LD       R11,^LAB(R1);       % (X)  MODE
43D6    342B 0014                  LD       R11,^(R2^(20));     % (BA) MODE
43DA    742B 0100                  LD       R11,^(R2^(R1));     % (BX) MODE
43DE                   %
43DE                   %
43DE                   % LOAD CONSTANT
43DE                   %
43DE    BD48                       LDK      R4,8;               % (R)  MODE
43E0                   %
43E0                               EJECT;
```



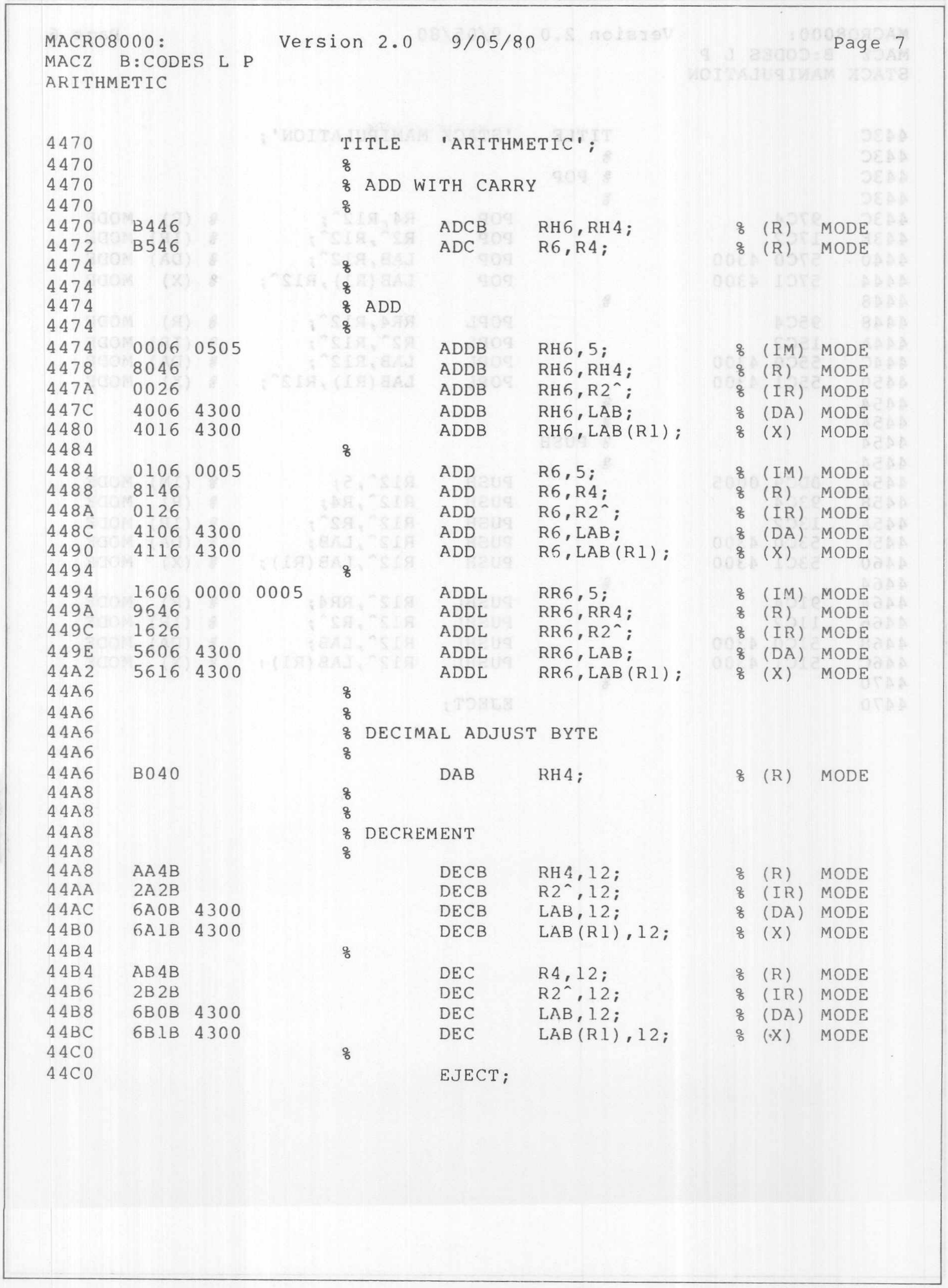
```
MACRO8000:              Version 2.0   9/05/80                     Page 4
MACZ  B:CODES L P
CLEAR, EXCHANGE, AND LOAD


43E0                       %
43E0                       % LOAD RELATIVE TO REGISTER
43E0                       %
43E0  3006 FF1C                    LDRB    RH6,LAB;       % (RA) MODE
43E4  3106 FF18                    LDR     R6,LAB;        % (RA) MODE
43E8  3506 FF14                    LDRL    RR6,LAB;       % (RA) MODE
43EC                       %
43EC                       %
43EC                       % LOAD RELATIVE TO MEMORY
43EC                       %
43EC  3206 FF10                    LDRB    LAB,RH6;       % (RA) MODE
43F0  3306 FF0C                    LDR     LAB,R6;        % (RA) MODE
43F4  3706 FF08                    LDRL    LAB,RR6;       % (RA) MODE
43F8                       %
43F8                       %
43F8                       %LOAD ADDRESS RELATIVE
43F8                       %
43F8  340B FF04                    LDR     R11,^LAB;      % (RA) MODE
43FC                       %
43FC                       %
43FC                       % LOAD MULTIPLE TO REGISTER
43FC                       %
43FC  1C21 0805                    LDM     R8,R2^,6;      % (IR) MODE
4400  5C01 0805 4300               LDM     R8,LAB,6;      % (DA) MODE
4406  5C11 0805 4300               LDM     R8,LAB(R1),6;  % (X)  MODE
440C                       %
440C                       %
440C                       % LOAD MULTIPLE TO MEMORY
440C                       %
440C  1C29 0805                    LDM     R2^,R8,6;      % (IR) MODE
4410  5C09 0805 4300               LDM     LAB,R8,6;      % (DA) MODE
4416  5C19 0805 4300               LDM     LAB(R1),R8,6;  % (X)  MODE
441C                       %
441C                       %
441C                       % LOAD AND DECREMENT
441C                       %
441C  BA29 0988                    LDDB    R8^,R2^,R9;    % (IR) MODE
4420  BB29 0988                    LDD     R8^,R2^,R9;    % (IR) MODE
4424                       %
4424                       %
4424                       % LOAD, DECREMENT, AND REPEAT
4424                       %
4424  BA29 0980                    LDDRB   R8^,R2^,R9;    % (IR) MODE
4428  BB29 0980                    LDDR    R8^,R2^,R9;    % (IR) MODE
442C                       %
442C                       %
442C                       % LOAD AND INCREMENT
442C                       %
442C  BA21 0988                    LDIB    R8^,R2^,R9;    % (IR) MODE
4430  BB21 0988                    LDI     R8^,R2^,R9;    % (IR) MODE
4434                       %
```



```
MACRO8000:          Version 2.0   9/05/80                          Page 5
MACZ  B:CODES L P
CLEAR, EXCHANGE, AND LOAD

4434                   %
4434                   % LOAD, INCREMENT, AND REPEAT
4434                   %
4434  BA21 0980              LDIRB   R8^,R2^,R9;     % (IR) MODE
4438  BB21 0980              LDIR    R8^,R2^,R9;     % (IR) MODE
443C                   %
443C                         EJECT;
```



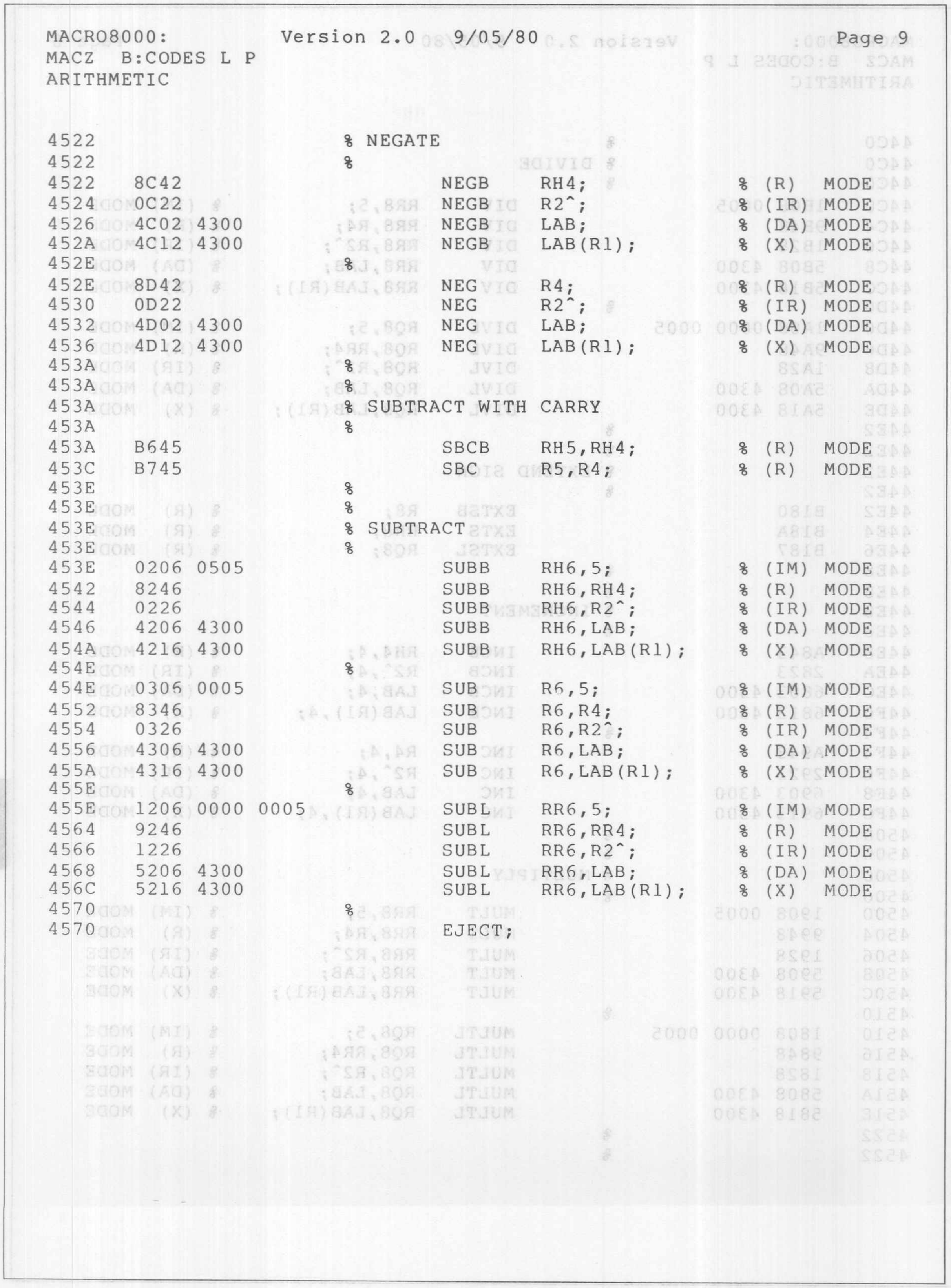
```
MACRO8000:          Version 2.0   9/05/80                    Page 6
MACZ  B:CODES L P
STACK MANIPULATION


443C                        TITLE   'STACK MANIPULATION';
443C                        %
443C                        % POP
443C                        %
443C    97C4                        POP     R4,R12^;         % (R)  MODE
443E    17C2                        POP     R2^,R12^;        % (IR) MODE
4440    57C0 4300                   POP     LAB,R12^;        % (DA) MODE
4444    57C1 4300                   POP     LAB(R1),R12^;    % (X)  MODE
4448                        %
4448    95C4                        POPL    RR4,R12^;        % (R)  MODE
444A    15C2                        POPL    R2^,R12^;        % (IR) MODE
444C    55C0 4300                   POPL    LAB,R12^;        % (DA) MODE
4450    55C1 4300                   POPL    LAB(R1),R12^;    % (X)  MODE
4454                        %
4454                        %
4454                        % PUSH
4454                        %
4454    0DC9 0005                   PUSH    R12^,5;          % (IM) MODE
4458    93C4                        PUSH    R12^,R4;         % (R)  MODE
445A    13C2                        PUSH    R12^,R2^;        % (IR) MODE
445C    53C0 4300                   PUSH    R12^,LAB;        % (DA) MODE
4460    53C1 4300                   PUSH    R12^,LAB(R1);    % (X)  MODE
4464                        %
4464    91C4                        PUSHL   R12^,RR4;        % (R)  MODE
4466    11C2                        PUSHL   R12^,R2^;        % (IR) MODE
4468    51C0 4300                   PUSHL   R12^,LAB;        % (DA) MODE
446C    51C1 4300                   PUSHL   R12^,LAB(R1);    % (X)  MODE
4470                        %
4470                                EJECT;
```



```
MACRO8000:            Version 2.0   9/05/80                    Page 7
MACZ  B:CODES L P
ARITHMETIC

4470                        TITLE  'ARITHMETIC';
4470                        %
4470                        % ADD WITH CARRY
4470                        %
4470  B446                          ADCB    RH6,RH4;           % (R)  MODE
4472  B546                          ADC     R6,R4;             % (R)  MODE
4474                        %
4474                        %
4474                        % ADD
4474                        %
4474  0006 0505                     ADDB    RH6,5;             % (IM) MODE
4478  8046                          ADDB    RH6,RH4;           % (R)  MODE
447A  0026                          ADDB    RH6,R2^;           % (IR) MODE
447C  4006 4300                     ADDB    RH6,LAB;           % (DA) MODE
4480  4016 4300                     ADDB    RH6,LAB(R1);       % (X)  MODE
4484                        %
4484  0106 0005                     ADD     R6,5;              % (IM) MODE
4488  8146                          ADD     R6,R4;             % (R)  MODE
448A  0126                          ADD     R6,R2^;            % (IR) MODE
448C  4106 4300                     ADD     R6,LAB;            % (DA) MODE
4490  4116 4300                     ADD     R6,LAB(R1);        % (X)  MODE
4494                        %
4494  1606 0000 0005                ADDL    RR6,5;             % (IM) MODE
449A  9646                          ADDL    RR6,RR4;           % (R)  MODE
449C  1626                          ADDL    RR6,R2^;           % (IR) MODE
449E  5606 4300                     ADDL    RR6,LAB;           % (DA) MODE
44A2  5616 4300                     ADDL    RR6,LAB(R1);       % (X)  MODE
44A6                        %
44A6                        %
44A6                        % DECIMAL ADJUST BYTE
44A6                        %
44A6  B040                          DAB     RH4;               % (R)  MODE
44A8                        %
44A8                        %
44A8                        % DECREMENT
44A8                        %
44A8  AA4B                          DECB    RH4,12;            % (R)  MODE
44AA  2A2B                          DECB    R2^,12;            % (IR) MODE
44AC  6A0B 4300                     DECB    LAB,12;            % (DA) MODE
44B0  6A1B 4300                     DECB    LAB(R1),12;        % (X)  MODE
44B4                        %
44B4  AB4B                          DEC     R4,12;             % (R)  MODE
44B6  2B2B                          DEC     R2^,12;            % (IR) MODE
44B8  6B0B 4300                     DEC     LAB,12;            % (DA) MODE
44BC  6B1B 4300                     DEC     LAB(R1),12;        % (X)  MODE
44C0                        %
44C0                                EJECT;
```

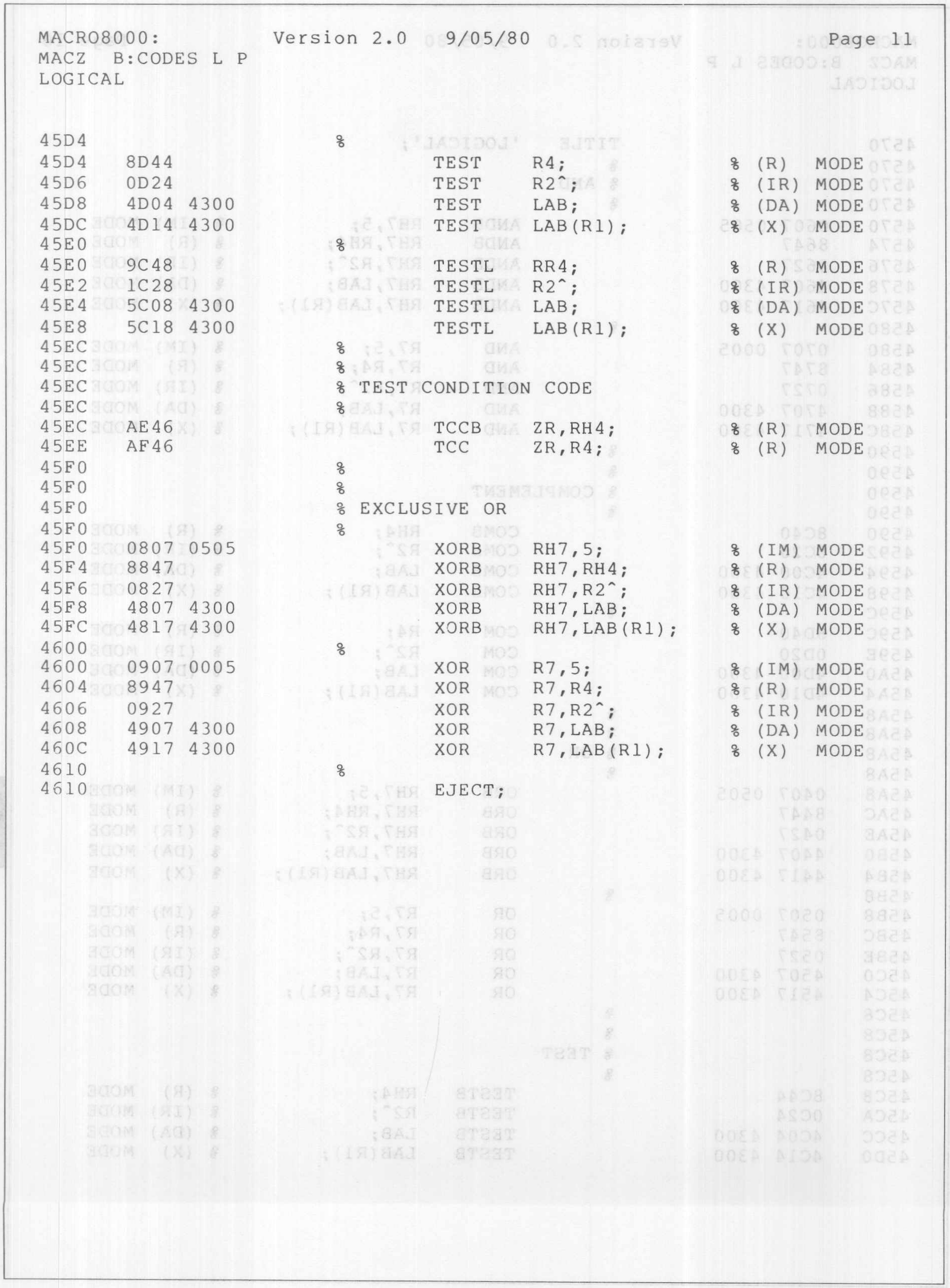
```
MACRO8000:         Version 2.0   9/05/80                  Page 8
MACZ  B:CODES L P
ARITHMETIC

44C0                    %
44C0                    % DIVIDE
44C0                    %
44C0   1B08 0005                DIV     RR8,5;           % (IM) MODE
44C4   9B48                     DIV     RR8,R4;          % (R)  MODE
44C6   1B28                     DIV     RR8,R2^;         % (IR) MODE
44C8   5B08 4300                DIV     RR8,LAB;         % (DA) MODE
44CC   5B18 4300                DIV     RR8,LAB(R1);     % (X)  MODE
44D0                    %
44D0   1A08 0000 0005           DIVL    RQ8,5;           % (IM) MODE
44D6   9A48                     DIVL    RQ8,RR4;         % (R)  MODE
44D8   1A28                     DIVL    RQ8,R2^;         % (IR) MODE
44DA   5A08 4300                DIVL    RQ8,LAB;         % (DA) MODE
44DE   5A18 4300                DIVL    RQ8,LAB(R1);     % (X)  MODE
44E2                    %
44E2                    %
44E2                    % EXTEND SIGN
44E2                    %
44E2   B180                     EXTSB   R8;              % (R)  MODE
44E4   B18A                     EXTS    RR8;             % (R)  MODE
44E6   B187                     EXTSL   RQ8;             % (R)  MODE
44E8                    %
44E8                    %
44E8                    % INCREMENT
44E8                    %
44E8   A843                     INCB    RH4,4;           % (R)  MODE
44EA   2823                     INCB    R2^,4;           % (IR) MODE
44EC   6803 4300                INCB    LAB,4;           % (DA) MODE
44F0   6813 4300                INCB    LAB(R1),4;       % (X)  MODE
44F4                    %
44F4   A943                     INC     R4,4;            % (R)  MODE
44F6   2923                     INC     R2^,4;           % (IR) MODE
44F8   6903 4300                INC     LAB,4;           % (DA) MODE
44FC   6913 4300                INC     LAB(R1),4;       % (X)  MODE
4500                    %
4500                    %
4500                    % MULTIPLY
4500                    %
4500   1908 0005                MULT    RR8,5;           % (IM) MODE
4504   9948                     MULT    RR8,R4;          % (R)  MODE
4506   1928                     MULT    RR8,R2^;         % (IR) MODE
4508   5908 4300                MULT    RR8,LAB;         % (DA) MODE
450C   5918 4300                MULT    RR8,LAB(R1);     % (X)  MODE
4510                    %
4510   1808 0000 0005           MULTL   RQ8,5;           % (IM) MODE
4516   9848                     MULTL   RQ8,RR4;         % (R)  MODE
4518   1828                     MULTL   RQ8,R2^;         % (IR) MODE
451A   5808 4300                MULTL   RQ8,LAB;         % (DA) MODE
451E   5818 4300                MULTL   RQ8,LAB(R1);     % (X)  MODE
4522                    %
4522                    %
```

A

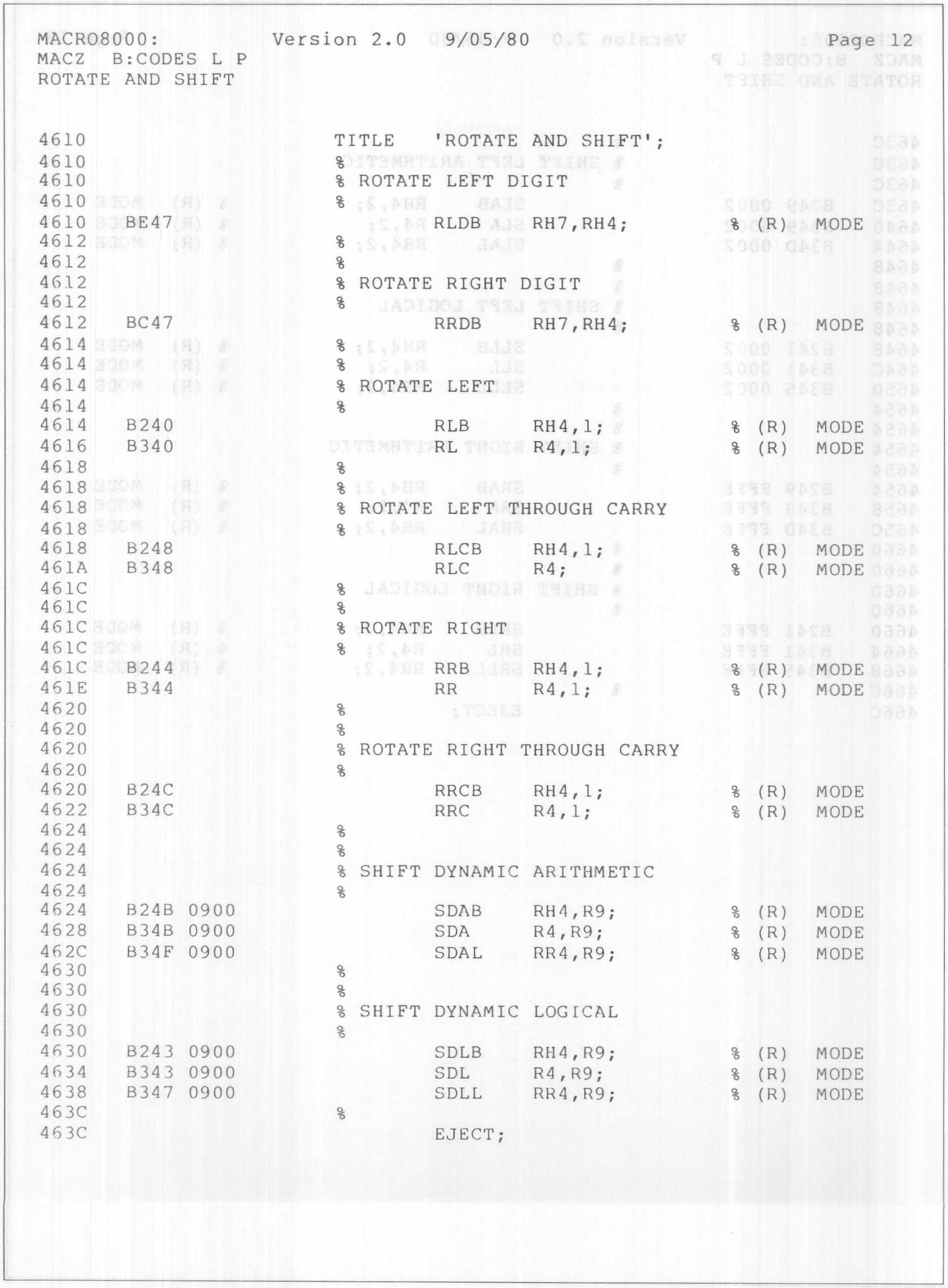


```
MACRO8000:          Version 2.0  9/05/80                   Page 9
MACZ  B:CODES L P
ARITHMETIC

4522                        % NEGATE
4522                        %
4522    8C42                         NEGB     RH4;              % (R)  MODE
4524    0C22                         NEGB     R2^;              % (IR) MODE
4526    4C02 4300                    NEGB     LAB;              % (DA) MODE
452A    4C12 4300                    NEGB     LAB(R1);          % (X)  MODE
452E                        %
452E    8D42                         NEG      R4;               % (R)  MODE
4530    0D22                         NEG      R2^;              % (IR) MODE
4532    4D02 4300                    NEG      LAB;              % (DA) MODE
4536    4D12 4300                    NEG      LAB(R1);          % (X)  MODE
453A                        %
453A                        %
453A                        % SUBTRACT WITH CARRY
453A                        %
453A    B645                         SBCB     RH5,RH4;          % (R)  MODE
453C    B745                         SBC      R5,R4;            % (R)  MODE
453E                        %
453E                        %
453E                        % SUBTRACT
453E                        %
453E    0206 0505                    SUBB     RH6,5;            % (IM) MODE
4542    8246                         SUBB     RH6,RH4;          % (R)  MODE
4544    0226                         SUBB     RH6,R2^;          % (IR) MODE
4546    4206 4300                    SUBB     RH6,LAB;          % (DA) MODE
454A    4216 4300                    SUBB     RH6,LAB(R1);      % (X)  MODE
454E                        %
454E    0306 0005                    SUB      R6,5;             % (IM) MODE
4552    8346                         SUB      R6,R4;            % (R)  MODE
4554    0326                         SUB      R6,R2^;           % (IR) MODE
4556    4306 4300                    SUB      R6,LAB;           % (DA) MODE
455A    4316 4300                    SUB      R6,LAB(R1);       % (X)  MODE
455E                        %
455E    1206 0000 0005               SUBL     RR6,5;            % (IM) MODE
4564    9246                         SUBL     RR6,RR4;          % (R)  MODE
4566    1226                         SUBL     RR6,R2^;          % (IR) MODE
4568    5206 4300                    SUBL     RR6,LAB;          % (DA) MODE
456C    5216 4300                    SUBL     RR6,LAB(R1);      % (X)  MODE
4570                        %
4570                                 EJECT;
```

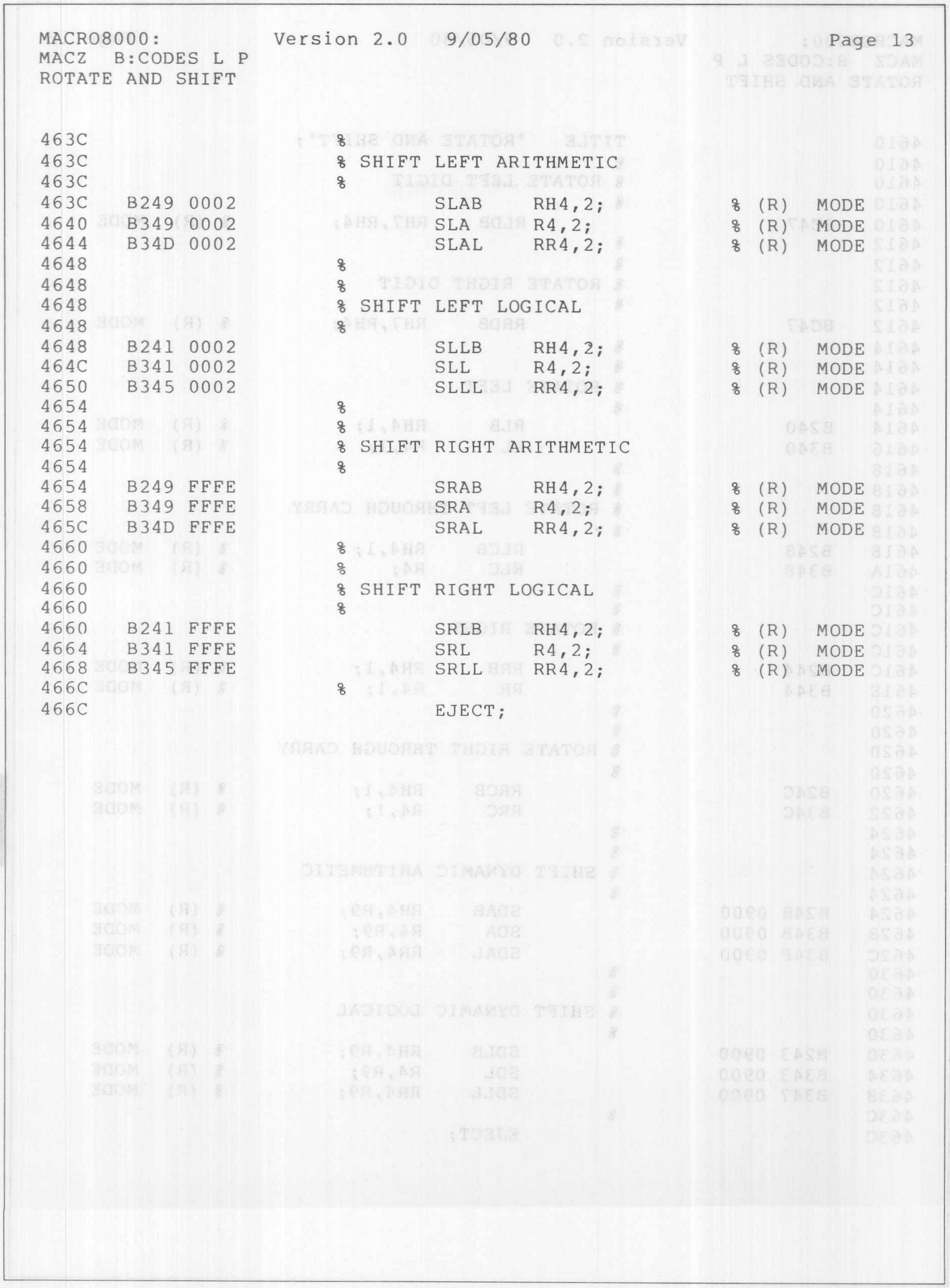
```
4570                    TITLE   'LOGICAL';
4570                    %
4570                    % AND
4570                    %
4570  0607 0505                 ANDB    RH7,5;          % (IM) MODE
4574   8647                     ANDB    RH7,RH4;        % (R)  MODE
4576  0627                      ANDB    RH7,R2^;        % (IR) MODE
4578  4607 4300                 ANDB    RH7,LAB;        % (DA) MODE
457C  4617 4300                 ANDB    RH7,LAB(R1);    % (X)  MODE
4580                    %
4580   0707 0005                AND     R7,5;           % (IM) MODE
4584   8747                     AND     R7,R4;          % (R)  MODE
4586   0727                     AND     R7,R2^;         % (IR) MODE
4588   4707 4300                AND     R7,LAB;         % (DA) MODE
458C   4717 4300                AND     R7,LAB(R1);     % (X)  MODE
4590                    %
4590                    %
4590                    % COMPLEMENT
4590                    %
4590   8C40                     COMB    RH4;            % (R)  MODE
4592   0C20                     COMB    R2^;            % (IR) MODE
4594   4C00 4300                COMB    LAB;            % (DA) MODE
4598   4C10 4300                COMB    LAB(R1);        % (X)  MODE
459C                    %
459C   8D40                     COM     R4;             % (R)  MODE
459E   0D20                     COM     R2^;            % (IR) MODE
45A0   4D00 4300                COM     LAB;            % (DA) MODE
45A4   4D10 4300                COM     LAB(R1);        % (X)  MODE
45A8                    %
45A8                    %
45A8                    % OR
45A8                    %
45A8   0407 0505                ORB     RH7,5;          % (IM) MODE
45AC   8447                     ORB     RH7,RH4;        % (R)  MODE
45AE   0427                     ORB     RH7,R2^;        % (IR) MODE
45B0   4407 4300                ORB     RH7,LAB;        % (DA) MODE
45B4   4417 4300                ORB     RH7,LAB(R1);    % (X)  MODE
45B8                    %
45B8   0507 0005                OR      R7,5;           % (IM) MODE
45BC   8547                     OR      R7,R4;          % (R)  MODE
45BE   0527                     OR      R7,R2^;         % (IR) MODE
45C0   4507 4300                OR      R7,LAB;         % (DA) MODE
45C4   4517 4300                OR      R7,LAB(R1);     % (X)  MODE
45C8                    %
45C8                    %
45C8                    % TEST
45C8                    %
45C8   8C44                     TESTB   RH4;            % (R)  MODE
45CA   0C24                     TESTB   R2^;            % (IR) MODE
45CC   4C04 4300                TESTB   LAB;            % (DA) MODE
45D0   4C14 4300                TESTB   LAB(R1);        % (X)  MODE
```



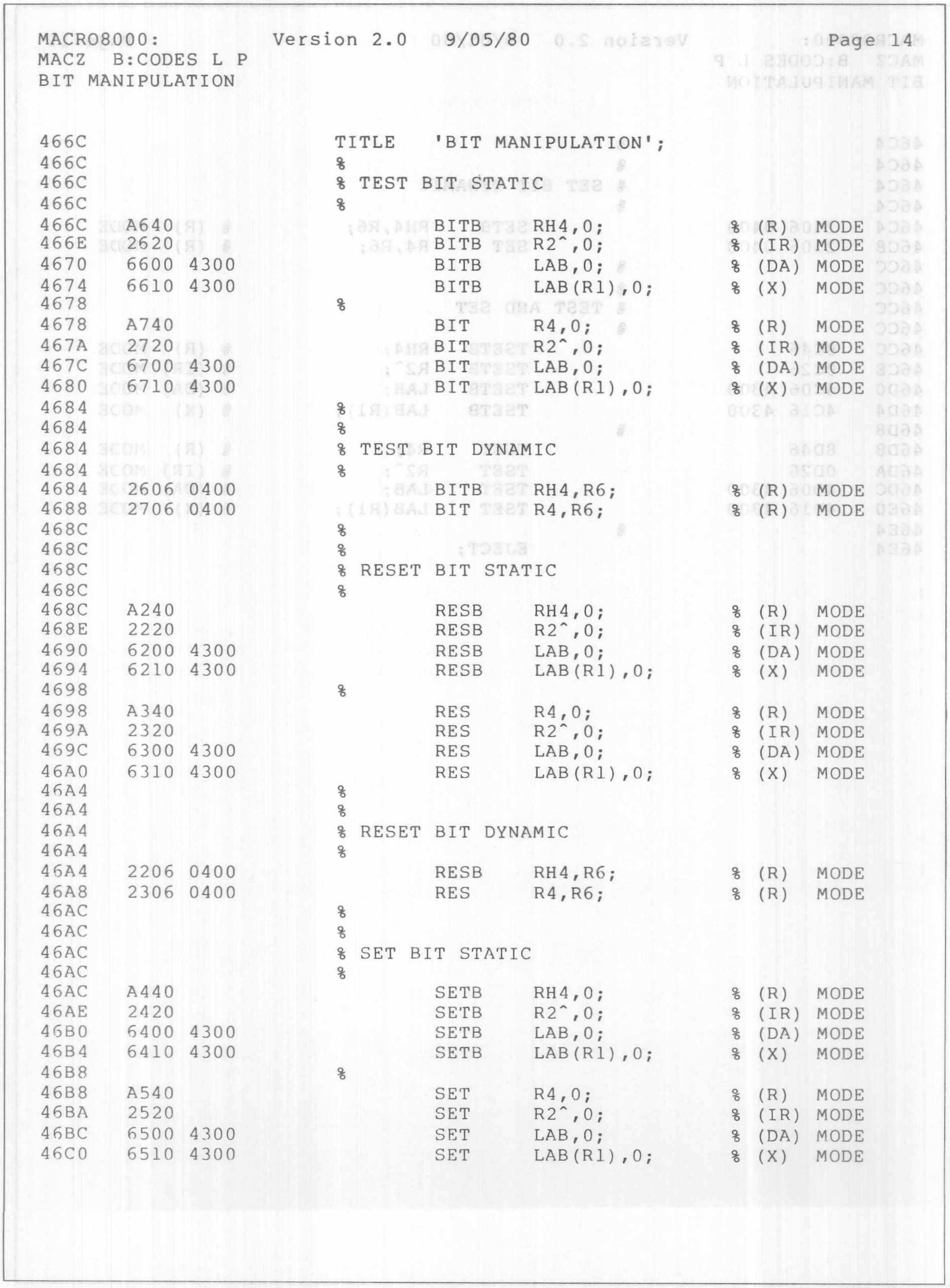
```
MACRO8000:          Version 2.0   9/05/80                             Page 11
MACZ  B:CODES L P
LOGICAL


45D4                     %
45D4   8D44                     TEST    R4;              % (R)  MODE
45D6   0D24                     TEST    R2^;             % (IR) MODE
45D8   4D04 4300                TEST    LAB;             % (DA) MODE
45DC   4D14 4300                TEST    LAB(R1);         % (X)  MODE
45E0                     %
45E0   9C48                     TESTL   RR4;             % (R)  MODE
45E2   1C28                     TESTL   R2^;             % (IR) MODE
45E4   5C08 4300                TESTL   LAB;             % (DA) MODE
45E8   5C18 4300                TESTL   LAB(R1);         % (X)  MODE
45EC                     %
45EC                     %
45EC                     % TEST CONDITION CODE
45EC                     %
45EC   AE46                     TCCB    ZR,RH4;          % (R)  MODE
45EE   AF46                     TCC     ZR,R4;           % (R)  MODE
45F0                     %
45F0                     %
45F0                     % EXCLUSIVE OR
45F0                     %
45F0   0807 0505                XORB    RH7,5;           % (IM) MODE
45F4   8847                     XORB    RH7,RH4;         % (R)  MODE
45F6   0827                     XORB    RH7,R2^;         % (IR) MODE
45F8   4807 4300                XORB    RH7,LAB;         % (DA) MODE
45FC   4817 4300                XORB    RH7,LAB(R1);     % (X)  MODE
4600                     %
4600   0907 0005                XOR     R7,5;            % (IM) MODE
4604   8947                     XOR     R7,R4;           % (R)  MODE
4606   0927                     XOR     R7,R2^;          % (IR) MODE
4608   4907 4300                XOR     R7,LAB;          % (DA) MODE
460C   4917 4300                XOR     R7,LAB(R1);      % (X)  MODE
4610                     %
4610                            EJECT;
```



```
MACRO8000:           Version 2.0  9/05/80                       Page 12
MACZ  B:CODES L P
ROTATE AND SHIFT


4610                           TITLE   'ROTATE AND SHIFT';
4610                           %
4610                           % ROTATE LEFT DIGIT
4610                           %
4610   BE47                           RLDB    RH7,RH4;            % (R)  MODE
4612                           %
4612                           %
4612                           % ROTATE RIGHT DIGIT
4612                           %
4612   BC47                           RRDB    RH7,RH4;            % (R)  MODE
4614                           %
4614                           %
4614                           % ROTATE LEFT
4614                           %
4614   B240                           RLB     RH4,1;              % (R)  MODE
4616   B340                           RL      R4,1;               % (R)  MODE
4618                           %
4618                           %
4618                           % ROTATE LEFT THROUGH CARRY
4618                           %
4618   B248                           RLCB    RH4,1;              % (R)  MODE
461A   B348                           RLC     R4;                 % (R)  MODE
461C                           %
461C                           %
461C                           % ROTATE RIGHT
461C                           %
461C   B244                           RRB     RH4,1;              % (R)  MODE
461E   B344                           RR      R4,1;               % (R)  MODE
4620                           %
4620                           %
4620                           % ROTATE RIGHT THROUGH CARRY
4620                           %
4620   B24C                           RRCB    RH4,1;              % (R)  MODE
4622   B34C                           RRC     R4,1;               % (R)  MODE
4624                           %
4624                           %
4624                           % SHIFT DYNAMIC ARITHMETIC
4624                           %
4624   B24B 0900                      SDAB    RH4,R9;             % (R)  MODE
4628   B34B 0900                      SDA     R4,R9;              % (R)  MODE
462C   B34F 0900                      SDAL    RR4,R9;             % (R)  MODE
4630                           %
4630                           %
4630                           % SHIFT DYNAMIC LOGICAL
4630                           %
4630   B243 0900                      SDLB    RH4,R9;             % (R)  MODE
4634   B343 0900                      SDL     R4,R9;              % (R)  MODE
4638   B347 0900                      SDLL    RR4,R9;             % (R)  MODE
463C                           %
463C                                  EJECT;
```



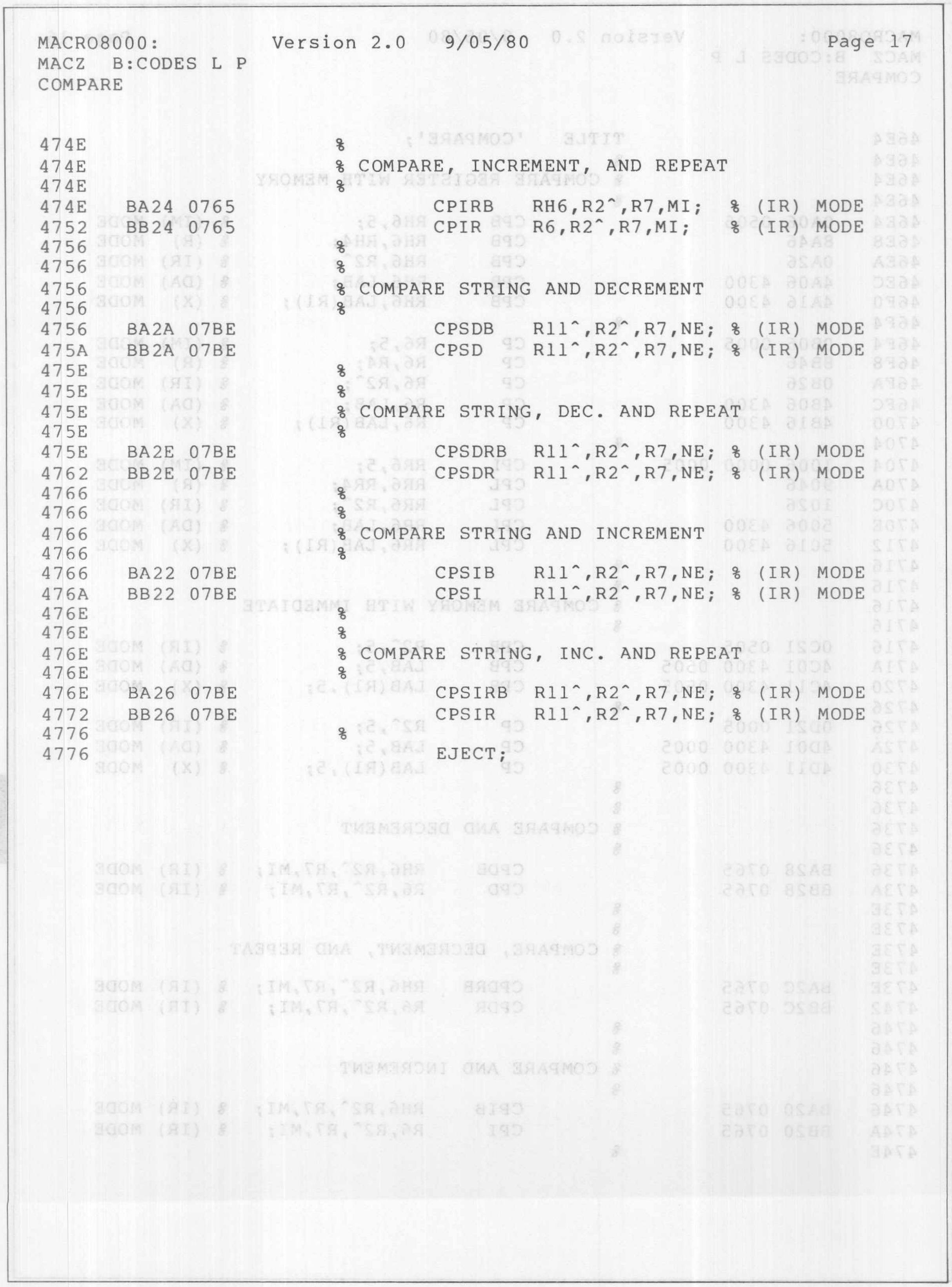
```
MACRO8000:          Version 2.0    9/05/80                          Page 13
MACZ  B:CODES L P
ROTATE AND SHIFT


463C                         %
463C                         % SHIFT LEFT ARITHMETIC
463C                         %
463C    B249 0002                    SLAB     RH4,2;         % (R)  MODE
4640    B349 0002                    SLA      R4,2;          % (R)  MODE
4644    B34D 0002                    SLAL     RR4,2;         % (R)  MODE
4648                         %
4648                         %
4648                         % SHIFT LEFT LOGICAL
4648                         %
4648    B241 0002                    SLLB     RH4,2;         % (R)  MODE
464C    B341 0002                    SLL      R4,2;          % (R)  MODE
4650    B345 0002                    SLLL     RR4,2;         % (R)  MODE
4654                         %
4654                         %
4654                         % SHIFT RIGHT ARITHMETIC
4654                         %
4654    B249 FFFE                    SRAB     RH4,2;         % (R)  MODE
4658    B349 FFFE                    SRA      R4,2;          % (R)  MODE
465C    B34D FFFE                    SRAL     RR4,2;         % (R)  MODE
4660                         %
4660                         %
4660                         % SHIFT RIGHT LOGICAL
4660                         %
4660    B241 FFFE                    SRLB     RH4,2;         % (R)  MODE
4664    B341 FFFE                    SRL      R4,2;          % (R)  MODE
4668    B345 FFFE                    SRLL     RR4,2;         % (R)  MODE
466C                         %
466C                                 EJECT;
```



```
MACRO8000:            Version 2.0   9/05/80                              Page 14
MACZ  B:CODES L P
BIT MANIPULATION

466C                        TITLE   'BIT MANIPULATION';
466C                        %
466C                        % TEST BIT STATIC
466C                        %
466C    A640                        BITB    RH4,0;              % (R)  MODE
466E    2620                        BITB    R2^,0;              % (IR) MODE
4670    6600 4300                   BITB    LAB,0;              % (DA) MODE
4674    6610 4300                   BITB    LAB(R1),0;          % (X)  MODE
4678                        %
4678    A740                        BIT     R4,0;               % (R)  MODE
467A    2720                        BIT     R2^,0;              % (IR) MODE
467C    6700 4300                   BIT     LAB,0;              % (DA) MODE
4680    6710 4300                   BIT     LAB(R1),0;          % (X)  MODE
4684                        %
4684                        %
4684                        % TEST BIT DYNAMIC
4684                        %
4684    2606 0400                   BITB    RH4,R6;             % (R)  MODE
4688    2706 0400                   BIT     R4,R6;              % (R)  MODE
468C                        %
468C                        %
468C                        % RESET BIT STATIC
468C                        %
468C    A240                        RESB    RH4,0;              % (R)  MODE
468E    2220                        RESB    R2^,0;              % (IR) MODE
4690    6200 4300                   RESB    LAB,0;              % (DA) MODE
4694    6210 4300                   RESB    LAB(R1),0;          % (X)  MODE
4698                        %
4698    A340                        RES     R4,0;               % (R)  MODE
469A    2320                        RES     R2^,0;              % (IR) MODE
469C    6300 4300                   RES     LAB,0;              % (DA) MODE
46A0    6310 4300                   RES     LAB(R1),0;          % (X)  MODE
46A4                        %
46A4                        %
46A4                        % RESET BIT DYNAMIC
46A4                        %
46A4    2206 0400                   RESB    RH4,R6;             % (R)  MODE
46A8    2306 0400                   RES     R4,R6;              % (R)  MODE
46AC                        %
46AC                        %
46AC                        % SET BIT STATIC
46AC                        %
46AC    A440                        SETB    RH4,0;              % (R)  MODE
46AE    2420                        SETB    R2^,0;              % (IR) MODE
46B0    6400 4300                   SETB    LAB,0;              % (DA) MODE
46B4    6410 4300                   SETB    LAB(R1),0;          % (X)  MODE
46B8                        %
46B8    A540                        SET     R4,0;               % (R)  MODE
46BA    2520                        SET     R2^,0;              % (IR) MODE
46BC    6500 4300                   SET     LAB,0;              % (DA) MODE
46C0    6510 4300                   SET     LAB(R1),0;          % (X)  MODE
```

```
MACRO8000:          Version 2.0  9/05/80                    Page 15
MACZ  B:CODES L P
BIT MANIPULATION


46C4                    %
46C4                    %
46C4                    % SET BIT DYNAMIC
46C4                    %
46C4  2406 0400               SETB    RH4,R6;          % (R)  MODE
46C8  2506 0400               SET     R4,R6;           % (R)  MODE
46CC                    %
46CC                    %
46CC                    % TEST AND SET
46CC                    %
46CC   8C46                   TSETB   RH4;             % (R)  MODE
46CE   0C26                   TSETB   R2^;             % (IR) MODE
46D0   4C06 4300              TSETB   LAB;             % (DA) MODE
46D4   4C16 4300              TSETB   LAB(R1);         % (X)  MODE
46D8                    %
46D8   8D46                   TSET    R4;              % (R)  MODE
46DA   0D26                   TSET    R2^;             % (IR) MODE
46DC   4D06 4300              TSET    LAB;             % (DA) MODE
46E0   4D16 4300              TSET    LAB(R1);         % (X)  MODE
46E4                    %
46E4                          EJECT;
```

```
46E4                    TITLE   'COMPARE';
46E4                    %
46E4                    % COMPARE REGISTER WITH MEMORY
46E4                    %
46E4   0A06 0505                CPB     RH6,5;          % (IM) MODE
46E8   8A46                     CPB     RH6,RH4;        % (R)  MODE
46EA   0A26                     CPB     RH6,R2^;        % (IR) MODE
46EC   4A06 4300                CPB     RH6,LAB;        % (DA) MODE
46F0   4A16 4300                CPB     RH6,LAB(R1);    % (X)  MODE
46F4                    %
46F4   0B06 0005                CP      R6,5;           % (IM) MODE
46F8   8B46                     CP      R6,R4;          % (R)  MODE
46FA   0B26                     CP      R6,R2^;         % (IR) MODE
46FC   4B06 4300                CP      R6,LAB;         % (DA) MODE
4700   4B16 4300                CP      R6,LAB(R1);     % (X)  MODE
4704                    %
4704   1006 0000 0005           CPL     RR6,5;          % (IM) MODE
470A   9046                     CPL     RR6,RR4;        % (R)  MODE
470C   1026                     CPL     RR6,R2^;        % (IR) MODE
470E   5006 4300                CPL     RR6,LAB;        % (DA) MODE
4712   5016 4300                CPL     RR6,LAB(R1);    % (X)  MODE
4716                    %
4716                    %
4716                    % COMPARE MEMORY WITH IMMEDIATE
4716                    %
4716   0C21 0505                CPB     R2^,5;          % (IR) MODE
471A   4C01 4300 0505           CPB     LAB,5;          % (DA) MODE
4720   4C11 4300 0505           CPB     LAB(R1),5;      % (X)  MODE
4726                    %
4726   0D21 0005                CP      R2^,5;          % (IR) MODE
472A   4D01 4300 0005           CP      LAB,5;          % (DA) MODE
4730   4D11 4300 0005           CP      LAB(R1),5;      % (X)  MODE
4736                    %
4736                    %
4736                    % COMPARE AND DECREMENT
4736                    %
4736   BA28 0765                CPDB    RH6,R2^,R7,MI;  % (IR) MODE
473A   BB28 0765                CPD     R6,R2^,R7,MI;   % (IR) MODE
473E                    %
473E                    %
473E                    % COMPARE, DECREMENT, AND REPEAT
473E                    %
473E   BA2C 0765                CPDRB   RH6,R2^,R7,MI;  % (IR) MODE
4742   BB2C 0765                CPDR    R6,R2^,R7,MI;   % (IR) MODE
4746                    %
4746                    %
4746                    % COMPARE AND INCREMENT
4746                    %
4746   BA20 0765                CPIB    RH6,R2^,R7,MI;  % (IR) MODE
474A   BB20 0765                CPI     R6,R2^,R7,MI;   % (IR) MODE
474E                    %
```



```
MACRO8000:          Version 2.0   9/05/80                  Page 17
MACZ  B:CODES L P
COMPARE

474E                         %
474E                         % COMPARE, INCREMENT, AND REPEAT
474E                         %
474E    BA24 0765                     CPIRB   RH6,R2^,R7,MI;  % (IR) MODE
4752    BB24 0765                     CPIR    R6,R2^,R7,MI;   % (IR) MODE
4756                         %
4756                         %
4756                         % COMPARE STRING AND DECREMENT
4756                         %
4756    BA2A 07BE                     CPSDB   R11^,R2^,R7,NE; % (IR) MODE
475A    BB2A 07BE                     CPSD    R11^,R2^,R7,NE; % (IR) MODE
475E                         %
475E                         %
475E                         % COMPARE STRING, DEC. AND REPEAT
475E                         %
475E    BA2E 07BE                     CPSDRB  R11^,R2^,R7,NE; % (IR) MODE
4762    BB2E 07BE                     CPSDR   R11^,R2^,R7,NE; % (IR) MODE
4766                         %
4766                         %
4766                         % COMPARE STRING AND INCREMENT
4766                         %
4766    BA22 07BE                     CPSIB   R11^,R2^,R7,NE; % (IR) MODE
476A    BB22 07BE                     CPSI    R11^,R2^,R7,NE; % (IR) MODE
476E                         %
476E                         %
476E                         % COMPARE STRING, INC. AND REPEAT
476E                         %
476E    BA26 07BE                     CPSIRB  R11^,R2^,R7,NE; % (IR) MODE
4772    BB26 07BE                     CPSIR   R11^,R2^,R7,NE; % (IR) MODE
4776                         %
4776                                  EJECT;
```

```
4776                       TITLE  'TRANSLATE';
4776                       %
4776                       % TRANSLATE AND DECREMENT
4776                       %
4776  B8B8 0620                   TRDB    R11^,R2^,R6;     % (IR) MODE
477A                       %
477A                       %
477A                       % TRANSLATE, DECREMENT, AND REPEAT
477A                       %
477A  B8BC 0620                   TRDRB   R11^,R2^,R6;     % (IR) MODE
477E                       %
477E                       %
477E                       % TRANSLATE AND INCREMENT
477E                       %
477E  B8B0 0620                   TRIB    R11^,R2^,R6;     % (IR) MODE
4782                       %
4782                       %
4782                       % TRANSLATE, INCREMENT AND REPEAT
4782                       %
4782  B8B4 0620                   TRIRB   R11^,R2^,R6;     % (IR) MODE
4786                       %
4786                       %
4786                       % TRANSLATE AND TEST, DECREMENT
4786                       %
4786  B8BA 0620                   TRTDB   R11^,R2^,R6;     % (IR) MODE
478A                       %
478A                       %
478A                       % TRANSLATE AND TEST, DEC. AND REPEAT
478A                       %
478A  B8BE 062E                   TRTDRB  R11^,R2^,R6;     % (IR) MODE
478E                       %
478E                       %
478E                       % TRANSLATE AND TEST, INCREMENT
478E                       %
478E  B8B2 0620                   TRTIB   R11^,R2^,R6;     % (IR) MODE
4792                       %
4792                       %
4792                       % TRANSLATE AND TEST, INC. AND REPEAT
4792                       %
4792  B8B6 062E                   TRTIRB  R11^,R2^,R6;     % (IR) MODE
4796                       %
4796                              EJECT;
```

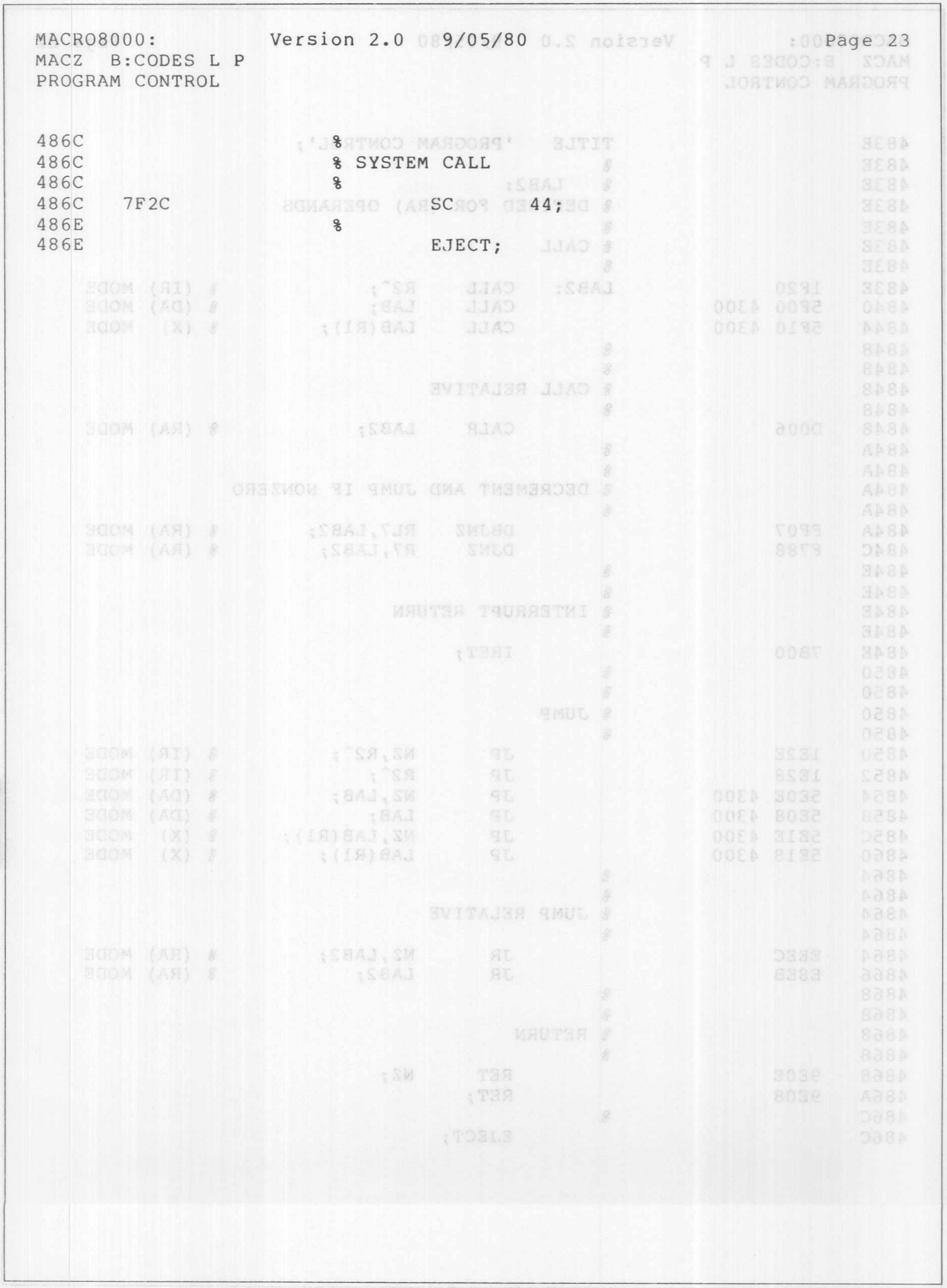
A

```
MACRO8000:           Version 2.0   9/05/80                  Page 19
MACZ  B:CODES L P
INPUT/OUTPUT


4796                     TITLE 'INPUT/OUTPUT';
4796                     %
4796                     % INPUT
4796                     %
4796   3CD4                     INB     RH4,R13;        % (PR) MODE
4798   3A44 0FC0                INB     RH4,#0FC0;      % (PA) MODE
479C                     %
479C   3DD4                     IN      R4,R13;         % (PR) MODE
479E   3B44 0FC0                IN      R4,#0FC0;       % (PA) MODE
47A2                     %
47A2                     %
47A2                     % INPUT AND DECREMENT
47A2                     %
47A2   3AD8 0928                INDB    R2^,R13,R9;     % (IR,PR) MODE
47A6   3BD8 0928                IND     R2^,R13,R9;     % (IR,PR) MODE
47AA                     %
47AA                     %
47AA                     % INPUT, DECREMENT AND REPEAT
47AA                     %
47AA   3AD8 0920                INDRB   R2^,R13,R9;     % (IR,PR) MODE
47AE   3BD8 0920                INDR    R2^,R13,R9;     % (IR,PR) MODE
47B2                     %
47B2                     %
47B2                     % INPUT AND INCREMENT
47B2                     %
47B2   3AD0 0928                INIB    R2^,R13,R9;     % (IR,PR) MODE
47B6   3BD0 0928                INI     R2^,R13,R9;     % (IR,PR) MODE
47BA                     %
47BA                     %
47BA                     % INPUT, INCREMENT AND REPEAT
47BA                     %
47BA   3AD0 0920                INIRB   R2^,R13,R9;     % (IR,PR) MODE
47BE   3BD0 0920                INIR    R2^,R13,R9;     % (IR,PR) MODE
47C2                     %
47C2                     %
47C2                     % OUTPUT
47C2                     %
47C2   3ED4                     OUTB    R13,RH4;        % (PR) MODE
47C4   3A46 0FC0                OUTB    #0FC0,RH4;      % (PA) MODE
47C8                     %
47C8   3FD4                     OUT     R13,R4;         % (PR) MODE
47CA   3B46 0FC0                OUT     #0FC0,R4;       % (PA) MODE
47CE                     %
47CE                     %
47CE                     % OUTPUT AND DECREMENT
47CE                     %
47CE   3A2A 09D8                OUTDB   R13,R2^,R9;     % (IR,PR) MODE
47D2   3B2A 09D8                OUTD    R13,R2^,R9;     % (IR,PR) MODE
47D6                     %
47D6                            EJECT;
```

```
47D6                     %
47D6                     % OUTPUT, DECREMENT AND REPEAT
47D6                     %
47D6    3A2A 09D0              OTDRB   R13,R2^,R9;       % (IR,PR) MODE
47DA    3B2A 09D0              OTDR    R13,R2^,R9;       % (IR,PR) MODE
47DE                     %
47DE                     %
47DE                     % OUTPUT AND INCREMENT
47DE                     %
47DE    3A22 09D8              OUTIB   R13,R2^,R9;       % (IR,PR) MODE
47E2    3B22 09D8              OUTI    R13,R2^,R9;       % (IR,PR) MODE
47E6                     %
47E6                     %
47E6                     % OUTPUT, INCREMENT AND REPEAT
47E6                     %
47E6    3A22 09D0              OTIRB   R13,R2^,R9;       % (IR,PR) MODE
47EA    3B22 09D0              OTIR    R13,R2^,R9;       % (IR,PR) MODE
47EE                     %
47EE                     %
47EE                     % SPECIAL INPUT
47EE                     %
47EE    3A45 0FC0              SINB    RH4,#0FC0;        % (PA) MODE
47F2    3B45 0FC0              SIN     R4,#0FC0;         % (PA) MODE
47F6                     %
47F6                     %
47F6                     % SPECIAL INPUT AND DECREMENT
47F6                     %
47F6    3AD9 0928              SINDB   R2^,R13,R9;       % (IR,PR) MODE
47FA    3BD9 0928              SIND    R2^,R13,R9;       % (IR,PR) MODE
47FE                     %
47FE                     %
47FE                     % SPECIAL INPUT, DECREMENT AND REPEAT
47FE                     %
47FE    3AD9 0920              SINDRB  R2^,R13,R9;       % (IR,PR) MODE
4802    3BD9 0920              SINDR   R2^,R13,R9;       % (IR,PR) MODE
4806                     %
4806                     %
4806                     % SPECIAL INPUT AND INCREMENT
4806                     %
4806    3AD1 0928              SINIB   R2^,R13,R9;       % (IR,PR) MODE
480A    3BD1 0928              SINI    R2^,R13,R9;       % (IR,PR) MODE
480E                     %
480E                     %
480E                     % SPECIAL INPUT, INCREMENT AND REPEAT
480E                     %
480E    3AD1 0920              SINIRB  R2^,R13,R9;       % (IR,PR) MODE
4812    3BD1 0920              SINIR   R2^,R13,R9;       % (IR,PR) MODE
4816                     %
4816                           EJECT;
```

```
MACRO8000:          Version 2.0   9/05/80                         Page 21
MACZ  B:CODES L P
INPUT/OUTPUT


4816                   %
4816                   % SPECIAL OUTPUT
4816                   %
4816   3A47 0FC0           SOUTB   #0FC0,RH4;      % (PA) MODE
481A   3B47 0FC0           SOUT    #0FC0,R4;       % (PA) MODE
481E                   %
481E                   %
481E                   % SPECIAL OUTPUT AND DECREMENT
481E                   %
481E   3A2B 09D8           SOUTDB  R13,R2^,R9;     % (IR,PR) MODE
4822   3B2B 09D8           SOUTD   R13,R2^,R9;     % (IR,PR) MODE
4826                   %
4826                   %
4826                   % SPECIAL OUTPUT, DECREMENT AND REPEAT
4826                   %
4826   3A2B 09D0           SOTDRB  R13,R2^,R9;     % (IR,PR) MODE
482A   3B2B 09D0           SOTDR   R13,R2^,R9;     % (IR,PR) MODE
482E                   %
482E                   %
482E                   % SPECIAL OUTPUT AND INCREMENT
482E                   %
482E   3A23 09D8           SOUTIB  R13,R2^,R9;     % (IR,PR) MODE
4832   3B23 09D8           SOUTI   R13,R2^,R9;     % (IR,PR) MODE
4836                   %
4836                   %
4836                   % SPECIAL OUTPUT, INCREMENT AND REPEAT
4836                   %
4836   3A23 09D0           SOTIRB  R13,R2^,R9;     % (IR,PR) MODE
483A   3B23 09D0           SOTIR   R13,R2^,R9;     % (IR,PR) MODE
483E                   %
483E                       EJECT;
```

```
483E                      TITLE   'PROGRAM CONTROL';
483E                      %
483E                      %   LAB2:
483E                      % DEFINED FOR (RA) OPERANDS
483E                      %
483E                      % CALL
483E                      %
483E  1F20                LAB2:   CALL    R2^;              % (IR) MODE
4840  5F00 4300                   CALL    LAB;              % (DA) MODE
4844  5F10 4300                   CALL    LAB(R1);          % (X)  MODE
4848                      %
4848                      %
4848                      % CALL RELATIVE
4848                      %
4848  D006                        CALR    LAB2;             % (RA) MODE
484A                      %
484A                      %
484A                      % DECREMENT AND JUMP IF NONZERO
484A                      %
484A  FF07                        DBJNZ   RL7,LAB2;         % (RA) MODE
484C  F788                        DJNZ    R7,LAB2;          % (RA) MODE
484E                      %
484E                      %
484E                      % INTERRUPT RETURN
484E                      %
484E  7B00                        IRET;
4850                      %
4850                      %
4850                      % JUMP
4850                      %
4850  1E2E                        JP      NZ,R2^;           % (IR) MODE
4852  1E28                        JP      R2^;              % (IR) MODE
4854  5E0E 4300                   JP      NZ,LAB;           % (DA) MODE
4858  5E08 4300                   JP      LAB;              % (DA) MODE
485C  5E1E 4300                   JP      NZ,LAB(R1);       % (X)  MODE
4860  5E18 4300                   JP      LAB(R1);          % (X)  MODE
4864                      %
4864                      %
4864                      % JUMP RELATIVE
4864                      %
4864  EEEC                        JR      NZ,LAB2;          % (RA) MODE
4866  E8EB                        JR      LAB2;             % (RA) MODE
4868                      %
4868                      %
4868                      % RETURN
4868                      %
4868  9E0E                        RET     NZ;
486A  9E08                        RET;
486C                      %
486C                              EJECT;
```



```
MACRO8000:          Version 2.0   9/05/80                          Page 23
MACZ  B:CODES L P
PROGRAM CONTROL


486C                        %
486C                        % SYSTEM CALL
486C                        %
486C    7F2C                      SC        44;
486E                        %
486E                              EJECT;
```

```
MACRO8000:         Version 2.0   9/05/80                      Page 24
MACZ  B:CODES L P
CPU CONTROL

486E                      TITLE   'CPU CONTROL';
486E                      %
486E                      % COMPLEMENT FLAGS
486E                      %
486E    8DC5                      COMFLG   CY,ZR;
4870                      %
4870                      %
4870                      % DISABLE INTERRUPT
4870                      %
4870    7C01                      DI       VI;
4872                      %
4872                      %
4872                      % ENABLE INTERRUPT
4872                      %
4872    7C04                      EI       NVI,VI;
4874                      %
4874                      %
4874                      % HALT
4874                      %
4874    7A00                      HALT;
4876                      %
4876                      %
4876                      % LOAD CONTROL REGISTER
4876                      %
4876    7DCA                      LDCTL    FCW,R12;        % (R)  MODE
4878                      %
4878                      %
4878                      % LOAD FROM CONTROL REGISTER
4878                      %
4878    7DC2                      LDCTL    R12,FCW;        % (R)  MODE
487A                      %
487A                      %
487A                      % LOAD FLAG BYTE
487A                      %
487A    8C79                      LDCTLB   FLAGS,RH7;      % (R)  MODE
487C                      %
487C                      %
487C                      % LOAD FROM FLAG BYTE
487C                      %
487C    8C71                      LDCTLB   RH7,FLAGS;      % (R)  MODE
487E                      %
487E                      %
487E                      % LOAD PROGRAM STATUS
487E                      %
487E    3920                      LDPS     R2^;            % (IR) MODE
4880    7900 4300                 LDPS     LAB;            % (DA) MODE
4884    7910 4300                 LDPS     LAB(R1);        % (X)  MODE
4888                      %
4888                              EJECT;%
```

```
MACRO8000:           Version 2.0    9/05/80                    Page 25
MACZ  B:CODES L P
CPU CONTROL


4888                    % MULTI-MICRO TEST
4888                    %
4888   7B0A                   MBIT;
488A                    %
488A                    %
488A                    % MULTI-MICRO REQUEST
488A                    %
488A   7BCD                   MREQ    R12;
488C                    %
488C                    %
488C                    % MULTI-MICRO RESET
488C                    %
488C   7B09                   MRES;
488E                    %
488E                    %
488E                    % MULTI-MICRO SET
488E                    %
488E   7B08                   MSET;
4890                    %
4890                    %
4890                    % NO OPERATION
4890                    %
4890   8D07                   NOP;
4892                    %
4892                    %
4892                    % RESET FLAGS
4892                    %
4892   8D43                   RESFLG  ZR;
4894                    %
4894                    %
4894                    % SET FLAGS
4894                    %
4894   8D71                   SETFLG  ZR,SGN,OV;
4896                    %
4896                    %
4896                          END.
```

APPENDIX B
AmZ8000 INSTRUCTION SET:
Numeric Listing by Opcode

```
0026                     ADDB    RH6,R2^;        % (IR) MODE
0106 0005                ADD     R6,5;           % (IM) MODE
0126                     ADD     R6,R2^;         % (IR) MODE
0206 0505                SUBB    RH6,5;          % (IM) MODE
0226                     SUBB    RH6,R2^;        % (IR) MODE
0306 0005                SUB     R6,5;           % (IM) MODE
0326                     SUB     R6,R2^;         % (IR) MODE
0407 0505                ORB     RH7,5;          % (IM) MODE
0427                     ORB     RH7,R2^;        % (IR) MODE
0507 0005                OR      R7,5;           % (IM) MODE
0527                     OR      R7,R2^;         % (IR) MODE
0607 0505                ANDB    RH7,5;          % (IM) MODE
0627                     ANDB    RH7,R2^;        % (IR) MODE
0707 0005                AND     R7,5;           % (IM) MODE
0727                     AND     R7,R2^;         % (IR) MODE
0807 0505                XORB    RH7,5;          % (IM) MODE
0827                     XORB    RH7,R2^;        % (IR) MODE
0907 0005                XOR     R7,5;           % (IM) MODE
0927                     XOR     R7,R2^;         % (IR) MODE
0A06 0505                CPB     RH6,5;          % (IM) MODE
0A26                     CPB     RH6,R2^;        % (IR) MODE
0B06 0005                CP      R6,5;           % (IM) MODE
0B26                     CP      R6,R2^;         % (IR) MODE
0C20                     COMB    R2^;            % (IR) MODE
0C21 0505                CPB     R2^,5;          % (IR) MODE
0C22                     NEGB    R2^;            % (IR) MODE
0C24                     TESTB   R2^;            % (IR) MODE
0C25 0505                LDB     R2^,5;          % (IR) MODE
0C26                     TSETB   R2^;            % (IR) MODE
0C28                     CLRB    R2^;            % (IR) MODE
0D20                     COM     R2^;            % (IR) MODE
0D21 0005                CP      R2^,5;          % (IR) MODE
0D22                     NEG     R2^;            % (IR) MODE
0D24                     TEST    R2^;            % (IR) MODE
0D25 0005                LD      R2^,5;          % (IR) MODE
0D26                     TSET    R2^;            % (IR) MODE
0D28                     CLR     R2^;            % (IR) MODE
0DC9 0005                PUSH    R12^,5;         % (IM) MODE

1006 0000 0005           CPL     RR6,5;          % (IM) MODE
1026                     CPL     RR6,R2^;        % (IR) MODE
11C2                     PUSHL   R12^,R2^;       % (IR) MODE
1206 0000 0005           SUBL    RR6,5;          % (IM) MODE
1226                     SUBL    RR6,R2^;        % (IR) MODE
13C2                     PUSH    R12^,R2^;       % (IR) MODE
1406 0000 0005           LDL     RR6,5;          % (IM) MODE
1426                     LDL     RR6,R2^;        % (IR) MODE
15C2                     POPL    R2^,R12^;       % (IR) MODE
1606 0000 0005           ADDL    RR6,5;          % (IM) MODE
1626                     ADDL    RR6,R2^;        % (IR) MODE
17C2                     POP     R2^,R12^;       % (IR) MODE
1808 0000 0005           MULTL   RQ8,5;          % (IM) MODE
1828                     MULTL   RQ8,R2^;        % (IR) MODE
1908 0005                MULT    RR8,5;          % (IM) MODE
1928                     MULT    RR8,R2^;        % (IR) MODE
1A08 0000 0005           DIVL    RQ8,5;          % (IM) MODE
```

```
1A28           DIVL    RQ8,R2^;          % (IR) MODE
1B08 0005      DIV     RR8,5;            % (IM) MODE
1B28           DIV     RR8,R2^;          % (IR) MODE
1C21 0805      LDM     R8,R2^,6;         % (IR) MODE
1C28           TESTL   R2^;              % (IR) MODE
1C29 0805      LDM     R2^,R8,6;         % (IR) MODE
1D26           LDL     R2^,RR6;          % (IR) MODE
1E28           JP      R2^;              % (IR) MODE
1E2E           JP      NZ,R2^;           % (IR) MODE
1F20     LAB2: CALL    R2^;              % (IR) MODE

2026           LDB     RH6,R2^;          % (IR) MODE
2106 0005      LD      R6,5;             % (IM) MODE
210B 4300      LD      R11,^LAB;         % (DA) MODE
2126           LD      R6,R2^;           % (IR) MODE
2206 0400      RESB    RH4,R6;           % (R) MODE
2220           RESB    R2^,0;            % (IR) MODE
2306 0400      RES     R4,R6;            % (R) MODE
2320           RES     R2^,0;            % (IR) MODE
2406 0400      SETB    RH4,R6;           % (R) MODE
2420           SETB    R2^,0;            % (IR) MODE
2506 0400      SET     R4,R6;            % (R) MODE
2520           SET     R2^,0;            % (IR) MODE
2606 0400      BITB    RH4,R6;           % (R) MODE
2620           BITB    R2^,0;            % (IR) MODE
2706 0400      BIT     R4,R6;            % (R) MODE
2720           BIT     R2^,0;            % (IR) MODE
2823           INCB    R2^,4;            % (IR) MODE
2923           INC     R2^,4;            % (IR) MODE
2A2B           DECB    R2^,12;           % (IR) MODE
2B2B           DEC     R2^,12;           % (IR) MODE
2C26           EXB     RH6,R2^;          % (IR) MODE
2D26           EX      R6,R2^;           % (IR) MODE
2E26           LDB     R2^,RH6;          % (IR) MODE
2F26           LD      R2^,R6;           % (IR) MODE

3006 FF1C      LDRB    RH6,LAB;          % (RA) MODE
3026 0014      LDB     RH6,R2^(20);      % (BA) MODE
3106 FF18      LDR     R6,LAB;           % (RA) MODE
3126 0014      LD      R6,R2^(20);       % (BA) MODE
3206 FF10      LDRB    LAB,RH6;          % (RA) MODE
3226 0014      LDB     R2^(20),RH6;      % (BA) MODE
3306 FF0C      LDR     LAB,R6;           % (RA) MODE
3326 0014      LD      R2^(20),R6;       % (BA) MODE
340B FF04      LDR     R11,^LAB;         % (RA) MODE
342B 0014      LD      R11,^(R2^(20));   % (BA) MODE
3506 FF14      LDRL    RR6,LAB;          % (RA) MODE
3526 0014      LDL     RR6,R2^(20);      % (BA) MODE
3706 FF08      LDRL    LAB,RR6;          % (RA) MODE
3726 0014      LDL     R2^(20),RR6;      % (BA) MODE
3920           LDPS    R2^;              % (IR) MODE
3A22 09D8      OUTIB   R13,R2^,R9;       % (IR,PR) MODE
3A22 09D0      OTIRB   R13,R2^,R9;       % (IR,PR) MODE
3A23 09D0      SOTIRB  R13,R2^,R9;       % (IR,PR) MODE
3A23 09D8      SOUTIB  R13,R2^,R9;       % (IR,PR) MODE
3A2A 09D8      OUTDB   R13,R2^,R9;       % (IR,PR) MODE
```

```
3A2A 09D0             OTDRB   R13,R2^,R9;       % (IR,PR) MODE
3A2B 09D8             SOUTDB  R13,R2^,R9;       % (IR,PR) MODE
3A2B 09D0             SOTDRB  R13,R2^,R9;       % (IR,PR) MODE
3A44 0FC0             INB     RH4,#0FC0;        % (PA) MODE
3A45 0FC0             SINB    RH4,#0FC0;        % (PA) MODE
3A46 0FC0             OUTB    #0FC0,RH4;        % (PA) MODE
3A47 0FC0             SOUTB   #0FC0,RH4;        % (PA) MODE
3AD0 0928             INIB    R2^,R13,R9;       % (IR,PR) MODE
3AD0 0920             INIRB   R2^,R13,R9;       % (IR,PR) MODE
3AD1 0928             SINIB   R2^,R13,R9;       % (IR,PR) MODE
3AD1 0920             SINIRB  R2^,R13,R9;       % (IR,PR) MODE
3AD8 0920             INDRB   R2^,R13,R9;       % (IR,PR) MODE
3AD8 0928             INDB    R2^,R13,R9;       % (IR,PR) MODE
3AD9 0920             SINDRB  R2^,R13,R9;       % (IR,PR) MODE
3AD9 0928             SINDB   R2^,R13,R9;       % (IR,PR) MODE
3B22 09D8             OUTI    R13,R2^,R9;       % (IR,PR) MODE
3B22 09D0             OTIR    R13,R2^,R9;       % (IR,PR) MODE
3B23 09D0             SOTIR   R13,R2^,R9;       % (IR,PR) MODE
3B23 09D8             SOUTI   R13,R2^,R9;       % (IR,PR) MODE
3B2A 09D8             OUTD    R13,R2^,R9;       % (IR,PR) MODE
3B2A 09D0             OTDR    R13,R2^,R9;       % (IR,PR) MODE
3B2B 09D0             SOTDR   R13,R2^,R9;       % (IR,PR) MODE
3B2B 09D8             SOUTD   R13,R2^,R9;       % (IR,PR) MODE
3B44 0FC0             IN      R4,#0FC0;         % (PA) MODE
3B45 0FC0             SIN     R4,#0FC0;         % (PA) MODE
3B46 0FC0             OUT     #0FC0,R4;         % (PA) MODE
3B47 0FC0             SOUT    #0FC0,R4;         % (PA) MODE
3BD0 0928             INI     R2^,R13,R9;       % (IR,PR) MODE
3BD0 0920             INIR    R2^,R13,R9;       % (IR,PR) MODE
3BD1 0920             SINIR   R2^,R13,R9;       % (IR,PR) MODE
3BD1 0928             SINI    R2^,R13,R9;       % (IR,PR) MODE
3BD8 0920             INDR    R2^,R13,R9;       % (IR,PR) MODE
3BD8 0928             IND     R2^,R13,R9;       % (IR,PR) MODE
3BD9 0920             SINDR   R2^,R13,R9;       % (IR,PR) MODE
3BD9 0928             SIND    R2^,R13,R9;       % (IR,PR) MODE
3CD4                  INB     RH4,R13;          % (PR) MODE
3DD4                  IN      R4,R13;           % (PR) MODE
3ED4                  OUTB    R13,RH4;          % (PR) MODE
3FD4                  OUT     R13,R4;           % (PR) MODE

4006 4300             ADDB    RH6,LAB;          % (DA) MODE
4016 4300             ADDB    RH6,LAB(R1);      % (X) MODE
4106 4300             ADD     R6,LAB;           % (DA) MODE
4116 4300             ADD     R6,LAB(R1);       % (X) MODE
4206 4300             SUBB    RH6,LAB;          % (DA) MODE
4216 4300             SUBB    RH6,LAB(R1);      % (X) MODE
4306 4300             SUB     R6,LAB;           % (DA) MODE
4316 4300             SUB     R6,LAB(R1);       % (X) MODE
4407 4300             ORB     RH7,LAB;          % (DA) MODE
4417 4300             ORB     RH7,LAB(R1);      % (X) MODE
4507 4300             OR      R7,LAB;           % (DA) MODE
4517 4300             OR      R7,LAB(R1);       % (X) MODE
4607 4300             ANDB    RH7,LAB;          % (DA) MODE
4617 4300             ANDB    RH7,LAB(R1);      % (X) MODE
4707 4300             AND     R7,LAB;           % (DA) MODE
4717 4300             AND     R7,LAB(R1);       % (X) MODE
```

```
4807 4300               XORB    RH7,LAB;        % (DA) MODE
4817 4300               XORB    RH7,LAB(R1);    % (X)  MODE
4907 4300               XOR     R7,LAB;         % (DA) MODE
4917 4300               XOR     R7,LAB(R1);     % (X)  MODE
4A06 4300               CPB     RH6,LAB;        % (DA) MODE
4A16 4300               CPB     RH6,LAB(R1);    % (X)  MODE
4B06 4300               CP      R6,LAB;         % (DA) MODE
4B16 4300               CP      R6,LAB(R1);     % (X)  MODE
4C00 4300               COMB    LAB;            % (DA) MODE
4C01 4300 0505          CPB     LAB,5;          % (DA) MODE
4C02 4300               NEGB    LAB;            % (DA) MODE
4C04 4300               TESTB   LAB;            % (DA) MODE
4C05 4300 0505          LDB     LAB,5;          % (DA) MODE
4C06 4300               TSETB   LAB;            % (DA) MODE
4C08 4300               CLRB    LAB;            % (DA) MODE
4C10 4300               COMB    LAB(R1);        % (X)  MODE
4C11 4300 0505          CPB     LAB(R1),5;      % (X)  MODE
4C12 4300               NEGB    LAB(R1);        % (X)  MODE
4C14 4300               TESTB   LAB(R1);        % (X)  MODE
4C15 4300 0505          LDB     LAB(R1),5;      % (X)  MODE
4C16 4300               TSETB   LAB(R1);        % (X)  MODE
4C18 4300               CLRB    LAB(R1);        % (X)  MODE
4D00 4300               COM     LAB;            % (DA) MODE
4D01 4300 0005          CP      LAB,5;          % (DA) MODE
4D02 4300               NEG     LAB;            % (DA) MODE
4D04 4300               TEST    LAB;            % (DA) MODE
4D05 4300 0005          LD      LAB,5;          % (DA) MODE
4D06 4300               TSET    LAB;            % (DA) MODE
4D08 4300               CLR     LAB;            % (DA) MODE
4D10 4300               COM     LAB(R1);        % (X)  MODE
4D11 4300 0005          CP      LAB(R1),5;      % (X)  MODE
4D12 4300               NEG     LAB(R1);        % (X)  MODE
4D14 4300               TEST    LAB(R1);        % (X)  MODE
4D15 4300 0005          LD      LAB(R1),5;      % (X)  MODE
4D16 4300               TSET    LAB(R1);        % (X)  MODE
4D18 4300               CLR     LAB(R1);        % (X)  MODE
5006 4300               CPL     RR6,LAB;        % (DA) MODE
5016 4300               CPL     RR6,LAB(R1);    % (X)  MODE
51C0 4300               PUSHL   R12^,LAB;       % (DA) MODE
51C1 4300               PUSHL   R12^,LAB(R1);   % (X)  MODE
5206 4300               SUBL    RR6,LAB;        % (DA) MODE
5216 4300               SUBL    RR6,LAB(R1);    % (X)  MODE
53C0 4300               PUSH    R12^,LAB;       % (DA) MODE
53C1 4300               PUSH    R12^,LAB(R1);   % (X)  MODE
5406 4300               LDL     RR6,LAB;        % (DA) MODE
5416 4300               LDL     RR6,LAB(R1);    % (X)  MODE
55C0 4300               POPL    LAB,R12^;       % (DA) MODE
55C1 4300               POPL    LAB(R1),R12^;   % (X)  MODE
5606 4300               ADDL    RR6,LAB;        % (DA) MODE
5616 4300               ADDL    RR6,LAB(R1);    % (X)  MODE
57C0 4300               POP     LAB,R12^;       % (DA) MODE
57C1 4300               POP     LAB(R1),R12^;   % (X)  MODE
5808 4300               MULTL   RQ8,LAB;        % (DA) MODE
5818 4300               MULTL   RQ8,LAB(R1);    % (X)  MODE
5908 4300               MULT    RR8,LAB;        % (DA) MODE
```

```
5918 4300                MULT    RR8,LAB(R1);    % (X)  MODE
5A08 4300                DIVL    RQ8,LAB;        % (DA) MODE
5A18 4300                DIVL    RQ8,LAB(R1);    % (X)  MODE
5B08 4300                DIV     RR8,LAB;        % (DA) MODE
5B18 4300                DIV     RR8,LAB(R1);    % (X)  MODE
5C01 0805 4300           LDM     R8,LAB,6;       % (DA) MODE
5C08 4300                TESTL   LAB;            % (DA) MODE
5C09 0805 4300           LDM     LAB,R8,6;       % (DA) MODE
5C11 0805 4300           LDM     R8,LAB(R1),6;   % (X)  MODE
5C18 4300                TESTL   LAB(R1);        % (X)  MODE
5C19 0805 4300           LDM     LAB(R1),R8,6;   % (X)  MODE
5D06 4300                LDL     LAB,RR6;        % (DA) MODE
5D16 4300                LDL     LAB(R1),RR6;    % (X)  MODE
5E08 4300                JP      LAB;            % (DA) MODE
5E0E 4300                JP      NZ,LAB;         % (DA) MODE
5E18 4300                JP      LAB(R1);        % (X)  MODE
5E1E 4300                JP      NZ,LAB(R1);     % (X)  MODE
5F00 4300                CALL    LAB;            % (DA) MODE
5F10 4300                CALL    LAB(R1);        % (X)  MODE
6006 4300                LDB     RH6,LAB;        % (DA) MODE
6016 4300                LDB     RH6,LAB(R1);    % (X)  MODE
6106 4300                LD      R6,LAB;         % (DA) MODE
6116 4300                LD      R6,LAB(R1);     % (X)  MODE
6200 4300                RESB    LAB,0;          % (DA) MODE
6210 4300                RESB    LAB(R1),0;      % (X)  MODE
6300 4300                RES     LAB,0;          % (DA) MODE
6310 4300                RES     LAB(R1),0;      % (X)  MODE
6400 4300                SETB    LAB,0;          % (DA) MODE
6410 4300                SETB    LAB(R1),0;      % (X)  MODE
6500 4300                SET     LAB,0;          % (DA) MODE
6510 4300                SET     LAB(R1),0;      % (X)  MODE
6600 4300                BITB    LAB,0;          % (DA) MODE
6610 4300                BITB    LAB(R1),0;      % (X)  MODE
6700 4300                BIT     LAB,0;          % (DA) MODE
6710 4300                BIT     LAB(R1),0;      % (X)  MODE
6803 4300                INCB    LAB,4;          % (DA) MODE
6813 4300                INCB    LAB(R1),4;      % (X)  MODE
6903 4300                INC     LAB,4;          % (DA) MODE
6913 4300                INC     LAB(R1),4;      % (X)  MODE
6A0B 4300                DECB    LAB,12;         % (DA) MODE
6A1B 4300                DECB    LAB(R1),12;     % (X)  MODE
6B0B 4300                DEC     LAB,12;         % (DA) MODE
6B1B 4300                DEC     LAB(R1),12;     % (X)  MODE
6C06 4300                EXB     RH6,LAB;        % (DA) MODE
6C16 4300                EXB     RH6,LAB(R1);    % (X)  MODE
6D06 4300                EX      R6,LAB;         % (DA) MODE
6D16 4300                EX      R6,LAB(R1);     % (X)  MODE
6E06 4300                LDB     LAB,RH6;        % (DA) MODE
6E16 4300                LDB     LAB(R1),RH6;    % (X)  MODE
6F06 4300                LD      LAB,R6;         % (DA) MODE
6F16 4300                LD      LAB(R1),R6;     % (X)  MODE
7026 0100                LDB     RH6,R2^(R1);    % (BX) MODE
7126 0100                LD      R6,R2^(R1);     % (BX) MODE
7226 0100                LDB     R2^(R1),RH6;    % (BX) MODE
```

```
7326 0100            LD      R2^(R1),R6;          % (BX) MODE
742B 0100            LD      R11,^(R2^(R1));      % (BX) MODE
7526 0100            LDL     RR6,R2^(R1);         % (BX) MODE
761B 4300            LD      R11,^LAB(R1);        % (X)  MODE
7726 0100            LDL     R2^(R1),RR6;         % (BX) MODE
7900 4300            LDPS    LAB;                 % (DA) MODE
7910 4300            LDPS    LAB(R1);             % (X)  MODE
7A00                 HALT;
7B00                 IRET;
7B08                 MSET;
7B09                 MRES;
7B0A                 MBIT;
7BCD                 MREQ    R12;
7C01                 DI      VI;
7C04                 EI      NVI,VI;
7DC2                 LDCTL   R12,FCW;             % (R)  MODE
7DCA                 LDCTL   FCW,R12;             % (R)  MODE
7F2C                 SC      44;
8046                 ADDB    RH6,RH4;             % (R)  MODE
8146                 ADD     R6,R4;               % (R)  MODE
8246                 SUBB    RH6,RH4;             % (R)  MODE
8346                 SUB     R6,R4;               % (R)  MODE
8447                 ORB     RH7,RH4;             % (R)  MODE
8547                 OR      R7,R4;               % (R)  MODE
8647                 ANDB    RH7,RH4;             % (R)  MODE
8747                 AND     R7,R4;               % (R)  MODE
8847                 XORB    RH7,RH4;             % (R)  MODE
8947                 XOR     R7,R4;               % (R)  MODE
8A46                 CPB     RH6,RH4;             % (R)  MODE
8B46                 CP      R6,R4;               % (R)  MODE
8C40                 COMB    RH4;                 % (R)  MODE
8C42                 NEGB    RH4;                 % (R)  MODE
8C44                 TESTB   RH4;                 % (R)  MODE
8C46                 TSETB   RH4;                 % (R)  MODE
8C48           LAB:  CLRB    RH4;                 % (R)  MODE
8C71                 LDCTLB  RH7,FLAGS;           % (R)  MODE
8C79                 LDCTLB  FLAGS,RH7;           % (R)  MODE
8D07                 NOP;
8D40                 COM     R4;                  % (R)  MODE
8D42                 NEG     R4;                  % (R)  MODE
8D43                 RESFLG  ZR;
8D44                 TEST    R4;                  % (R)  MODE
8D46                 TSET    R4;                  % (R)  MODE
8D48                 CLR     R4;                  % (R)  MODE
8D71                 SETFLG  ZR,SGN,OV;
8DC5                 COMFLG  CY,ZR;
9046                 CPL     RR6,RR4;             % (R)  MODE
91C4                 PUSHL   R12^,RR4;            % (R)  MODE
9246                 SUBL    RR6,RR4;             % (R)  MODE
93C4                 PUSH    R12^,R4;             % (R)  MODE
9446                 LDL     RR6,RR4;             % (R)  MODE
95C4                 POPL    RR4,R12^;            % (R)  MODE
9646                 ADDL    RR6,RR4;             % (R)  MODE
97C4                 POP     R4,R12^;             % (R)  MODE
```

```
9848                 MULTL   RQ8,RR4;              % (R)  MODE
9948                 MULT    RR8,R4;               % (R)  MODE
9A48                 DIVL    RQ8,RR4;              % (R)  MODE
9B48                 DIV     RR8,R4;               % (R)  MODE
9C48                 TESTL   RR4;                  % (R)  MODE
9E08                 RET;
9E0E                 RET     NZ;

A046                 LDB     RH6,RH4;              % (R)  MODE
A146                 LD      R6,R4;                % (R)  MODE
A240                 RESB    RH4,0;                % (R)  MODE
A340                 RES     R4,0;                 % (R)  MODE
A440                 SETB    RH4,0;                % (R)  MODE
A540                 SET     R4,0;                 % (R)  MODE
A640                 BITB    RH4,0;                % (R)  MODE
A740                 BIT     R4,0;                 % (R)  MODE
A843                 INCB    RH4,4;                % (R)  MODE
A943                 INC     R4,4;                 % (R)  MODE
AA4B                 DECB    RH4,12;               % (R)  MODE
AB4B                 DEC     R4,12;                % (R)  MODE
AC46                 EXB     RH6,RH4;              % (R)  MODE
AD46                 EX      R6,R4;                % (R)  MODE
AE46                 TCCB    ZR,RH4;               % (R)  MODE
AF46                 TCC     ZR,R4;                % (R)  MODE

B040                 DAB     RH4;                  % (R)  MODE
B180                 EXTSB   R8;                   % (R)  MODE
B187                 EXTSL   RQ8;                  % (R)  MODE
B18A                 EXTS    RR8;                  % (R)  MODE
B240                 RLB     RH4,1;                % (R)  MODE
B241 FFFE            SRLB    RH4,2;                % (R)  MODE
B241 0002            SLLB    RH4,2;                % (R)  MODE
B243 0900            SDLB    RH4,R9;               % (R)  MODE
B244                 RRB     RH4,1;                % (R)  MODE
B248                 RLCB    RH4,1;                % (R)  MODE
B249 0002            SLAB    RH4,2;                % (R)  MODE
B249 FFFE            SRAB    RH4,2;                % (R)  MODE
B24B 0900            SDAB    RH4,R9;               % (R)  MODE
B24C                 RRCB    RH4,1;                % (R)  MODE
B340                 RL      R4,1;                 % (R)  MODE
B341 0002            SLL     R4,2;                 % (R)  MODE
B341 FFFE            SRL     R4,2;                 % (R)  MODE
B343 0900            SDL     R4,R9;                % (R)  MODE
B344                 RR      R4,1;                 % (R)  MODE
B345 FFFE            SRLL    RR4,2;                % (R)  MODE
B345 0002            SLLL    RR4,2;                % (R)  MODE
B347 0900            SDLL    RR4,R9;               % (R)  MODE
B348                 RLC     R4;                   % (R)  MODE
B349 FFFE            SRA     R4,2;                 % (R)  MODE
B349 0002            SLA     R4,2;                 % (R)  MODE
B34B 0900            SDA     R4,R9;                % (R)  MODE
B34C                 RRC     R4,1;                 % (R)  MODE
B34D 0002            SLAL    RR4,2;                % (R)  MODE
B34D FFFE            SRAL    RR4,2;                % (R)  MODE
B34F 0900            SDAL    RR4,R9;               % (R)  MODE
B446                 ADCB    RH6,RH4;              % (R)  MODE
```

```
B546                ADC     R6,R4;              % (R)  MODE
B645                SBCB    RH5,RH4;            % (R)  MODE
B745                SBC     R5,R4;              % (R)  MODE
B8B0 0620           TRIB    R11^,R2^,R6;        % (IR) MODE
B8B2 0620           TRTIB   R11^,R2^,R6;        % (IR) MODE
B8B4 0620           TRIRB   R11^,R2^,R6;        % (IR) MODE
B8B6 062E           TRTIRB  R11^,R2^,R6;        % (IR) MODE
B8B8 0620           TRDB    R11^,R2^,R6;        % (IR) MODE
B8BA 0620           TRTDB   R11^,R2^,R6;        % (IR) MODE
B8BC 0620           TRDRB   R11^,R2^,R6;        % (IR) MODE
B8BE 062E           TRTDRB  R11^,R2^,R6;        % (IR) MODE
BA20 0765           CPIB    RH6,R2^,R7,MI;      % (IR) MODE
BA21 0988           LDIB    R8^,R2^,R9;         % (IR) MODE
BA21 0980           LDIRB   R8^,R2^,R9;         % (IR) MODE
BA22 07BE           CPSIB   R11^,R2^,R7,NE;     % (IR) MODE
BA24 0765           CPIRB   RH6,R2^,R7,MI;      % (IR) MODE
BA26 07BE           CPSIRB  R11^,R2^,R7,NE;     % (IR) MODE
BA28 0765           CPDB    RH6,R2^,R7,MI;      % (IR) MODE
BA29 0988           LDDB    R8^,R2^,R9;         % (IR) MODE
BA29 0980           LDDRB   R8^,R2^,R9;         % (IR) MODE
BA2A 07BE           CPSDB   R11^,R2^,R7,NE;     % (IR) MODE
BA2C 0765           CPDRB   RH6,R2^,R7,MI;      % (IR) MODE
BA2E 07BE           CPSDRB  R11^,R2^,R7,NE;     % (IR) MODE
BB20 0765           CPI     R6,R2^,R7,MI;       % (IR) MODE
BB21 0980           LDIR    R8^,R2^,R9;         % (IR) MODE
BB21 0988           LDI     R8^,R2^,R9;         % (IR) MODE
BB22 07BE           CPSI    R11^,R2^,R7,NE;     % (IR) MODE
BB24 0765           CPIR    R6,R2^,R7,MI;       % (IR) MODE
BB26 07BE           CPSIR   R11^,R2^,R7,NE;     % (IR) MODE
BB28 0765           CPD     R6,R2^,R7,MI;       % (IR) MODE
BB29 0980           LDDR    R8^,R2^,R9;         % (IR) MODE
BB29 0988           LDD     R8^,R2^,R9;         % (IR) MODE
BB2A 07BE           CPSD    R11^,R2^,R7,NE;     % (IR) MODE
BB2C 0765           CPDR    R6,R2^,R7,MI;       % (IR) MODE
BB2E 07BE           CPSDR   R11^,R2^,R7,NE;     % (IR) MODE
BC47                RRDB    RH7,RH4;            % (R)  MODE
BD48                LDK     R4,8;               % (R)  MODE
BE47                RLDB    RH7,RH4;            % (R)  MODE

C605                LDB     RH6,5;              % (IM) MODE

D006                CALR    LAB2;               % (RA) MODE

E8EB                JR      LAB2;               % (RA) MODE
EEEC                JR      NZ,LAB2;            % (RA) MODE

F788                DJNZ    R7,LAB2;            % (RA) MODE
FF07                DBJNZ   RL7,LAB2;           % (RA) MODE
```

APPENDIX C
AmZ8000 INSTRUCTION SET:
Alphabetic Listing by Mnemonic

```
4106 4300               ADD     R6,LAB;          % (DA) MODE
4116 4300               ADD     R6,LAB(R1);      % (X)  MODE
0106 0005               ADD     R6,5;            % (IM) MODE
0126                    ADD     R6,R2^;          % (IR) MODE
8046                    ADDB    RH6,RH4;         % (R)  MODE
4006 4300               ADDB    RH6,LAB;         % (DA) MODE
4016 4300               ADDB    RH6,LAB(R1);     % (X)  MODE
0026                    ADDB    RH6,R2^;         % (IR) MODE
9646                    ADDL    RR6,RR4;         % (R)  MODE
1606 0000 0005          ADDL    RR6,5;           % (IM) MODE
1626                    ADDL    RR6,R2^;         % (IR) MODE
5606 4300               ADDL    RR6,LAB;         % (DA) MODE
5616 4300               ADDL    RR6,LAB(R1);     % (X)  MODE
8747                    AND     R7,R4;           % (R)  MODE
4707 4300               AND     R7,LAB;          % (DA) MODE
4717 4300               AND     R7,LAB(R1);      % (X)  MODE
0707 0005               AND     R7,5;            % (IM) MODE
0727                    AND     R7,R2^;          % (IR) MODE
8647                    ANDB    RH7,RH4;         % (R)  MODE
4607 4300               ANDB    RH7,LAB;         % (DA) MODE
4617 4300               ANDB    RH7,LAB(R1);     % (X)  MODE
0607 0505               ANDB    RH7,5;           % (IM) MODE
0627                    ANDB    RH7,R2^;         % (IR) MODE

2706 0400               BIT     R4,R6;           % (R)  MODE
2720                    BIT     R2^,0;           % (IR) MODE
A740                    BIT     R4,0;            % (R)  MODE
6700 4300               BIT     LAB,0;           % (DA) MODE
6710 4300               BIT     LAB(R1),0;       % (X)  MODE
2606 0400               BITB    RH4,R6;          % (R)  MODE
2620                    BITB    R2^,0;           % (IR) MODE
A640                    BITB    RH4,0;           % (R)  MODE
6600 4300               BITB    LAB,0;           % (DA) MODE
6610 4300               BITB    LAB(R1),0;       % (X)  MODE

1F20            LAB2:   CALL    R2^;             % (IR) MODE
5F00 4300               CALL    LAB;             % (DA) MODE
5F10 4300               CALL    LAB(R1);         % (X)  MODE
D006                    CALR    LAB2;            % (RA) MODE
0D28                    CLR     R2^;             % (IR) MODE
8D48                    CLR     R4;              % (R)  MODE
4D08 4300               CLR     LAB;             % (DA) MODE
4D18 4300               CLR     LAB(R1);         % (X)  MODE
8C48            LAB:    CLRB    RH4;             % (R)  MODE
4C08 4300               CLRB    LAB;             % (DA) MODE
4C18 4300               CLRB    LAB(R1);         % (X)  MODE
0C28                    CLRB    R2^;             % (IR) MODE
8D40                    COM     R4;              % (R)  MODE
4D00 4300               COM     LAB;             % (DA) MODE
4D10 4300               COM     LAB(R1);         % (X)  MODE
0D20                    COM     R2^;             % (IR) MODE
8C40                    COMB    RH4;             % (R)  MODE
4C00 4300               COMB    LAB;             % (DA) MODE
```

```
4C10 4300            COMB    LAB(R1);            % (X)  MODE
0C20                 COMB    R2^;                % (IR) MODE
8DC5                 COMFLG  CY,ZR;
0D21 0005            CP      R2^,5;              % (IR) MODE
8B46                 CP      R6,R4;              % (R)  MODE
4B06 4300            CP      R6,LAB;             % (DA) MODE
4B16 4300            CP      R6,LAB(R1);         % (X)  MODE
0B06 0005            CP      R6,5;               % (IM) MODE
4D01 4300 0005       CP      LAB,5;              % (DA) MODE
0B26                 CP      R6,R2^;             % (IR) MODE
4D11 4300 0005       CP      LAB(R1),5;          % (X)  MODE
8A46                 CPB     RH6,RH4;            % (R)  MODE
0A06 0505            CPB     RH6,5;              % (IM) MODE
4A06 4300            CPB     RH6,LAB;            % (DA) MODE
4A16 4300            CPB     RH6,LAB(R1);        % (X)  MODE
0A26                 CPB     RH6,R2^;            % (IR) MODE
4C01 4300 0505       CPB     LAB,5;              % (DA) MODE
4C11 4300 0505       CPB     LAB(R1),5;          % (X)  MODE
0C21 0505            CPB     R2^,5;              % (IR) MODE
BB28 0765            CPD     R6,R2^,R7,MI;       % (IR) MODE
BA28 0765            CPDB    RH6,R2^,R7,MI;      % (IR) MODE
BB2C 0765            CPDR    R6,R2^,R7,MI;       % (IR) MODE
BA2C 0765            CPDRB   RH6,R2^,R7,MI;      % (IR) MODE
BB20 0765            CPI     R6,R2^,R7,MI;       % (IR) MODE
BA20 0765            CPIB    RH6,R2^,R7,MI;      % (IR) MODE
BB24 0765            CPIR    R6,R2^,R7,MI;       % (IR) MODE
BA24 0765            CPIRB   RH6,R2^,R7,MI;      % (IR) MODE
1006 0000 0005       CPL     RR6,5;              % (IM) MODE
9046                 CPL     RR6,RR4;            % (R)  MODE
1026                 CPL     RR6,R2^;            % (IR) MODE
5006 4300            CPL     RR6,LAB;            % (DA) MODE
5016 4300            CPL     RR6,LAB(R1);        % (X)  MODE
BB2A 07BE            CPSD    R11^,R2^,R7,NE;     % (IR) MODE
BA2A 07BE            CPSDB   R11^,R2^,R7,NE;     % (IR) MODE
BB2E 07BE            CPSDR   R11^,R2^,R7,NE;     % (IR) MODE
BA2E 07BE            CPSDRB  R11^,R2^,R7,NE;     % (IR) MODE
BB22 07BE            CPSI    R11^,R2^,R7,NE;     % (IR) MODE
BA22 07BE            CPSIB   R11^,R2^,R7,NE;     % (IR) MODE
BB26 07BE            CPSIR   R11^,R2^,R7,NE;     % (IR) MODE
BA26 07BE            CPSIRB  R11^,R2^,R7,NE;     % (IR) MODE

B040                 DAB     RH4;                % (R)  MODE
FF07                 DBJNZ   RL7,LAB2;           % (RA) MODE
6B0B 4300            DEC     LAB,12;             % (DA) MODE
6B1B 4300            DEC     LAB(R1),12;         % (X)  MODE
2B2B                 DEC     R2^,12;             % (IR) MODE
AB4B                 DEC     R4,12;              % (R)  MODE
6A0B 4300            DECB    LAB,12;             % (DA) MODE
6A1B 4300            DECB    LAB(R1),12;         % (X)  MODE
2A2B                 DECB    R2^,12;             % (IR) MODE
AA4B                 DECB    RH4,12;             % (R)  MODE
7C01                 DI      VI;
9B48                 DIV     RR8,R4;             % (R)  MODE
5B08 4300            DIV     RR8,LAB;            % (DA) MODE
5B18 4300            DIV     RR8,LAB(R1);        % (X)  MODE
1B08 0005            DIV     RR8,5;              % (IM) MODE
```

```
1B28                     DIV     RR8,R2^;         % (IR) MODE
9A48                     DIVL    RQ8,RR4;         % (R)  MODE
5A08 4300                DIVL    RQ8,LAB;         % (DA) MODE
1A08 0000 0005           DIVL    RQ8,5;           % (IM) MODE
5A18 4300                DIVL    RQ8,LAB(R1);     % (X)  MODE
1A28                     DIVL    RQ8,R2^;         % (IR) MODE
F788                     DJNZ    R7,LAB2;         % (RA) MODE

7C04                     EI      NVI,VI;
6D06 4300                EX      R6,LAB;          % (DA) MODE
6D16 4300                EX      R6,LAB(R1);      % (X)  MODE
2D26                     EX      R6,R2^;          % (IR) MODE
AD46                     EX      R6,R4;           % (R)  MODE
6C06 4300                EXB     RH6,LAB;         % (DA) MODE
6C16 4300                EXB     RH6,LAB(R1);     % (X)  MODE
2C26                     EXB     RH6,R2^;         % (IR) MODE
AC46                     EXB     RH6,RH4;         % (R)  MODE
B18A                     EXTS    RR8;             % (R)  MODE
B180                     EXTSB   R8;              % (R)  MODE
B187                     EXTSL   RQ8;             % (R)  MODE

7A00                     HALT;

3B44 0FC0                IN      R4,#0FC0;        % (PA) MODE
3DD4                     IN      R4,R13;          % (PR) MODE
3CD4                     INB     RH4,R13;         % (PR) MODE
3A44 0FC0                INB     RH4,#0FC0;       % (PA) MODE
6903 4300                INC     LAB,4;           % (DA) MODE
6913 4300                INC     LAB(R1),4;       % (X)  MODE
2923                     INC     R2^,4;           % (IR) MODE
A943                     INC     R4,4;            % (R)  MODE
6813 4300                INCB    LAB(R1),4;       % (X)  MODE
2823                     INCB    R2^,4;           % (IR) MODE
A843                     INCB    RH4,4;           % (R)  MODE
6803 4300                INCB    LAB,4;           % (DA) MODE
3BD8 0928                IND     R2^,R13,R9;      % (IR,PR) MODE
3AD8 0928                INDB    R2^,R13,R9;      % (IR,PR) MODE
3BD8 0920                INDR    R2^,R13,R9;      % (IR,PR) MODE
3AD8 0920                INDRB   R2^,R13,R9;      % (IR,PR) MODE
3BD0 0928                INI     R2^,R13,R9;      % (IR,PR) MODE
3AD0 0928                INIB    R2^,R13,R9;      % (IR,PR) MODE
3BD0 0920                INIR    R2^,R13,R9;      % (IR,PR) MODE
3AD0 0920                INIRB   R2^,R13,R9;      % (IR,PR) MODE
7B00                     IRET;

1E2E                     JP      NZ,R2^;          % (IR) MODE
5E08 4300                JP      LAB;             % (DA) MODE
5E0E 4300                JP      NZ,LAB;          % (DA) MODE
5E18 4300                JP      LAB(R1);         % (X)  MODE
5E1E 4300                JP      NZ,LAB(R1);      % (X)  MODE
1E28                     JP      R2^;             % (IR) MODE
E8EB                     JR      LAB2;            % (RA) MODE
EEEC                     JR      NZ,LAB2;         % (RA) MODE

2106 0005                LD      R6,5;            % (IM) MODE
6F06 4300                LD      LAB,R6;          % (DA) MODE
```

```
6F16 4300           LD      LAB(R1),R6;        % (X)  MODE
7126 0100           LD      R6,R2^(R1);        % (BX) MODE
210B 4300           LD      R11,^LAB;          % (DA) MODE
7326 0100           LD      R2^(R1),R6;        % (BX) MODE
742B 0100           LD      R11,^(R2^(R1));    % (BX) MODE
2126                LD      R6,R2^;            % (IR) MODE
761B 4300           LD      R11,^LAB(R1);      % (X)  MODE
0D25 0005           LD      R2^,5;             % (IR) MODE
342B 0014           LD      R11,^(R2^(20));    % (BA) MODE
A146                LD      R6,R4;             % (R)  MODE
2F26                LD      R2^,R6;            % (IR) MODE
3126 0014           LD      R6,R2^(20);        % (BA) MODE
4D05 4300 0005      LD      LAB,5;             % (DA) MODE
3326 0014           LD      R2^(20),R6;        % (BA) MODE
6106 4300           LD      R6,LAB;            % (DA) MODE
6116 4300           LD      R6,LAB(R1);        % (X)  MODE
4D15 4300 0005      LD      LAB(R1),5;         % (X)  MODE
2026                LDB     RH6,R2^;           % (IR) MODE
6E06 4300           LDB     LAB,RH6;           % (DA) MODE
6E16 4300           LDB     LAB(R1),RH6;       % (X)  MODE
7026 0100           LDB     RH6,R2^(R1);       % (BX) MODE
7226 0100           LDB     R2^(R1),RH6;       % (BX) MODE
A046                LDB     RH6,RH4;           % (R)  MODE
2E26                LDB     R2^,RH6;           % (IR) MODE
4C05 4300 0505      LDB     LAB,5;             % (DA) MODE
3026 0014           LDB     RH6,R2^(20);       % (BA) MODE
4C15 4300 0505      LDB     LAB(R1),5;         % (X)  MODE
3226 0014           LDB     R2^(20),RH6;       % (BA) MODE
C605                LDB     RH6,5;             % (IM) MODE
0C25 0505           LDB     R2^,5;             % (IR) MODE
6006 4300           LDB     RH6,LAB;           % (DA) MODE
6016 4300           LDB     RH6,LAB(R1);       % (X)  MODE
7DC2                LDCTL   R12,FCW;           % (R)  MODE
7DCA                LDCTL   FCW,R12;           % (R)  MODE
8C71                LDCTLB  RH7,FLAGS;         % (R)  MODE
8C79                LDCTLB  FLAGS,RH7;         % (R)  MODE
BB29 0988           LDD     R8^,R2^,R9;        % (IR) MODE
BA29 0988           LDDB    R8^,R2^,R9;        % (IR) MODE
BB29 0980           LDDR    R8^,R2^,R9;        % (IR) MODE
BA29 0980           LDDRB   R8^,R2^,R9;        % (IR) MODE
BB21 0988           LDI     R8^,R2^,R9;        % (IR) MODE
BA21 0988           LDIB    R8^,R2^,R9;        % (IR) MODE
BB21 0980           LDIR    R8^,R2^,R9;        % (IR) MODE
BA21 0980           LDIRB   R8^,R2^,R9;        % (IR) MODE
BD48                LDK     R4,8;              % (R)  MODE
7526 0100           LDL     RR6,R2^(R1);       % (BX) MODE
7726 0100           LDL     R2^(R1),RR6;       % (BX) MODE
5416 4300           LDL     RR6,LAB(R1);       % (X)  MODE
5406 4300           LDL     RR6,LAB;           % (DA) MODE
9446                LDL     RR6,RR4;           % (R)  MODE
3526 0014           LDL     RR6,R2^(20);       % (BA) MODE
1406 0000 0005      LDL     RR6,5;             % (IM) MODE
1426                LDL     RR6,R2^;           % (IR) MODE
3726 0014           LDL     R2^(20),RR6;       % (BA) MODE
5D06 4300           LDL     LAB,RR6;           % (DA) MODE
5D16 4300           LDL     LAB(R1),RR6;       % (X)  MODE
```

A

```
1D26                      LDL     R2^,RR6;        % (IR) MODE
5C01 0805 4300            LDM     R8,LAB,6;       % (DA) MODE
5C09 0805 4300            LDM     LAB,R8,6;       % (DA) MODE
5C11 0805 4300            LDM     R8,LAB(R1),6;   % (X)  MODE
5C19 0805 4300            LDM     LAB(R1),R8,6;   % (X)  MODE
1C21 0805                 LDM     R8,R2^,6;       % (IR) MODE
1C29 0805                 LDM     R2^,R8,6;       % (IR) MODE
7900 4300                 LDPS    LAB;            % (DA) MODE
7910 4300                 LDPS    LAB(R1);        % (X)  MODE
3920                      LDPS    R2^;            % (IR) MODE
3106 FF18                 LDR     R6,LAB;         % (RA) MODE
3306 FF0C                 LDR     LAB,R6;         % (RA) MODE
340B FF04                 LDR     R11,^LAB;       % (RA) MODE
3006 FF1C                 LDRB    RH6,LAB;        % (RA) MODE
3206 FF10                 LDRB    LAB,RH6;        % (RA) MODE
3706 FF08                 LDRL    LAB,RR6;        % (RA) MODE
3506 FF14                 LDRL    RR6,LAB;        % (RA) MODE

7B0A                      MBIT;
7BCD                      MREQ    R12;
7B09                      MRES;
7B08                      MSET;
1908 0005                 MULT    RR8,5;          % (IM) MODE
9948                      MULT    RR8,R4;         % (R)  MODE
1928                      MULT    RR8,R2^;        % (IR) MODE
5908 4300                 MULT    RR8,LAB;        % (DA) MODE
5918 4300                 MULT    RR8,LAB(R1);    % (X)  MODE
1828                      MULTL   RQ8,R2^;        % (IR) MODE
1808 0000 0005            MULTL   RQ8,5;          % (IM) MODE
9848                      MULTL   RQ8,RR4;        % (R)  MODE
5808 4300                 MULTL   RQ8,LAB;        % (DA) MODE
5818 4300                 MULTL   RQ8,LAB(R1);    % (X)  MODE

0D22                      NEG     R2^;            % (IR) MODE
4D12 4300                 NEG     LAB(R1);        % (X)  MODE
8D42                      NEG     R4;             % (R)  MODE
4D02 4300                 NEG     LAB;            % (DA) MODE
8C42                      NEGB    RH4;            % (R)  MODE
4C02 4300                 NEGB    LAB;            % (DA) MODE
4C12 4300                 NEGB    LAB(R1);        % (X)  MODE
0C22                      NEGB    R2^;            % (IR) MODE
8D07                      NOP;

8547                      OR      R7,R4;          % (R)  MODE
4507 4300                 OR      R7,LAB;         % (DA) MODE
4517 4300                 OR      R7,LAB(R1);     % (X)  MODE
0507 0005                 OR      R7,5;           % (IM) MODE
0527                      OR      R7,R2^;         % (IR) MODE
8447                      ORB     RH7,RH4;        % (R)  MODE
0407 0505                 ORB     RH7,5;          % (IM) MODE
0427                      ORB     RH7,R2^;        % (IR) MODE
4407 4300                 ORB     RH7,LAB;        % (DA) MODE
4417 4300                 ORB     RH7,LAB(R1);    % (X)  MODE
3B2A 09D0                 OTDR    R13,R2^,R9;     % (IR,PR) MODE
3A2A 09D0                 OTDRB   R13,R2^,R9;     % (IR,PR) MODE
3B22 09D0                 OTIR    R13,R2^,R9;     % (IR,PR) MODE
```

```
3A22 09D0          OTIRB   R13,R2^,R9;       % (IR,PR) MODE
3FD4               OUT     R13,R4;           % (PR) MODE
3B46 0FC0          OUT     #0FC0,R4;         % (PA) MODE
3A46 0FC0          OUTB    #0FC0,RH4;        % (PA) MODE
3ED4               OUTB    R13,RH4;          % (PR) MODE
3B2A 09D8          OUTD    R13,R2^,R9;       % (IR,PR) MODE
3A2A 09D8          OUTDB   R13,R2^,R9;       % (IR,PR) MODE
3B22 09D8          OUTI    R13,R2^,R9;       % (IR,PR) MODE
3A22 09D8          OUTIB   R13,R2^,R9;       % (IR,PR) MODE

97C4               POP     R4,R12^;          % (R)  MODE
17C2               POP     R2^,R12^;         % (IR) MODE
57C0 4300          POP     LAB,R12^;         % (DA) MODE
57C1 4300          POP     LAB(R1),R12^;     % (X)  MODE
95C4               POPL    RR4,R12^;         % (R)  MODE
55C1 4300          POPL    LAB(R1),R12^;     % (X)  MODE
55C0 4300          POPL    LAB,R12^;         % (DA) MODE
15C2               POPL    R2^,R12^;         % (IR) MODE
0DC9 0005          PUSH    R12^,5;           % (IM) MODE
53C1 4300          PUSH    R12^,LAB(R1);     % (X)  MODE
93C4               PUSH    R12^,R4;          % (R)  MODE
53C0 4300          PUSH    R12^,LAB;         % (DA) MODE
13C2               PUSH    R12^,R2^;         % (IR) MODE
11C2               PUSHL   R12^,R2^;         % (IR) MODE
51C1 4300          PUSHL   R12^,LAB(R1);     % (X)  MODE
91C4               PUSHL   R12^,RR4;         % (R)  MODE
51C0 4300          PUSHL   R12^,LAB;         % (DA) MODE

2306 0400          RES     R4,R6;            % (R)  MODE
2320               RES     R2^,0;            % (IR) MODE
A340               RES     R4,0;             % (R)  MODE
6300 4300          RES     LAB,0;            % (DA) MODE
6310 4300          RES     LAB(R1),0;        % (X)  MODE
2206 0400          RESB    RH4,R6;           % (R)  MODE
A240               RESB    RH4,0;            % (R)  MODE
6200 4300          RESB    LAB,0;            % (DA) MODE
6210 4300          RESB    LAB(R1),0;        % (X)  MODE
2220               RESB    R2^,0;            % (IR) MODE
8D43               RESFLG  ZR;
9E0E               RET     NZ;
9E08               RET;
B340               RL      R4,1;             % (R)  MODE
B240               RLB     RH4,1;            % (R)  MODE
B348               RLC     R4,1;             % (R)  MODE
B248               RLCB    RH4,1;            % (R)  MODE
BE47               RLDB    RH7,RH4;          % (R)  MODE
B344               RR      R4,1;             % (R)  MODE
B244               RRB     RH4,1;            % (R)  MODE
B34C               RRC     R4,1;             % (R)  MODE
B24C               RRCB    RH4,1;            % (R)  MODE
BC47               RRDB    RH7,RH4;          % (R)  MODE

B745               SBC     R5,R4;            % (R)  MODE
B645               SBCB    RH5,RH4;          % (R)  MODE
7F2C               SC      44;
B34B 0900          SDA     R4,R9;            % (R)  MODE
```

```
B24B 0900           SDAB    RH4,R9;             % (R)  MODE
B34F 0900           SDAL    RR4,R9;             % (R)  MODE
B343 0900           SDL     R4,R9;              % (R)  MODE
B243 0900           SDLB    RH4,R9;             % (R)  MODE
B347 0900           SDLL    RR4,R9;             % (R)  MODE
2520                SET     R2^,0;              % (IR) MODE
2506 0400           SET     R4,R6;              % (R)  MODE
6510 4300           SET     LAB(R1),0;          % (X)  MODE
A540                SET     R4,0;               % (R)  MODE
6500 4300           SET     LAB,0;              % (DA) MODE
6400 4300           SETB    LAB,0;              % (DA) MODE
6410 4300           SETB    LAB(R1),0;          % (X)  MODE
2406 0400           SETB    RH4,R6;             % (R)  MODE
2420                SETB    R2^,0;              % (IR) MODE
A440                SETB    RH4,0;              % (R)  MODE
8D71                SETFLG  ZR,SGN,OV;
3B45 0FC0           SIN     R4,#0FC0;           % (PA) MODE
3A45 0FC0           SINB    RH4,#0FC0;          % (PA) MODE
3BD9 0928           SIND    R2^,R13,R9;         % (IR,PR) MODE
3AD9 0928           SINDB   R2^,R13,R9;         % (IR,PR) MODE
3BD9 0920           SINDR   R2^,R13,R9;         % (IR,PR) MODE
3AD9 0920           SINDRB  R2^,R13,R9;         % (IR,PR) MODE
3BD1 0928           SINI    R2^,R13,R9;         % (IR,PR) MODE
3AD1 0928           SINIB   R2^,R13,R9;         % (IR,PR) MODE
3BD1 0920           SINIR   R2^,R13,R9;         % (IR,PR) MODE
3AD1 0920           SINIRB  R2^,R13,R9;         % (IR,PR) MODE
B349 0002           SLA     R4,2;               % (R)  MODE
B249 0002           SLAB    RH4,2;              % (R)  MODE
B34D 0002           SLAL    RR4,2;              % (R)  MODE
B341 0002           SLL     R4,2;               % (R)  MODE
B241 0002           SLLB    RH4,2;              % (R)  MODE
B345 0002           SLLL    RR4,2;              % (R)  MODE
3B2B 09D0           SOTDR   R13,R2^,R9;         % (IR,PR) MODE
3A2B 09D0           SOTDRB  R13,R2^,R9;         % (IR,PR) MODE
3B23 09D0           SOTIR   R13,R2^,R9;         % (IR,PR) MODE
3A23 09D0           SOTIRB  R13,R2^,R9;         % (IR,PR) MODE
3B47 0FC0           SOUT    #0FC0,R4;           % (PA) MODE
3A47 0FC0           SOUTB   #0FC0,RH4;          % (PA) MODE
3B2B 09D8           SOUTD   R13,R2^,R9;         % (IR,PR) MODE
3A2B 09D8           SOUTDB  R13,R2^,R9;         % (IR,PR) MODE
3B23 09D8           SOUTI   R13,R2^,R9;         % (IR,PR) MODE
3A23 09D8           SOUTIB  R13,R2^,R9;         % (IR,PR) MODE
B349 FFFE           SRA     R4,2;               % (R)  MODE
B249 FFFE           SRAB    RH4,2;              % (R)  MODE
B34D FFFE           SRAL    RR4,2;              % (R)  MODE
B341 FFFE           SRL     R4,2;               % (R)  MODE
B241 FFFE           SRLB    RH4,2;              % (R)  MODE
B345 FFFE           SRLL    RR4,2;              % (R)  MODE
4306 4300           SUB     R6,LAB;             % (DA) MODE
8346                SUB     R6,R4;              % (R)  MODE
0326                SUB     R6,R2^;             % (IR) MODE
0306 0005           SUB     R6,5;               % (IM) MODE
4316 4300           SUB     R6,LAB(R1);         % (X)  MODE
8246                SUBB    RH6,RH4;            % (R)  MODE
0206 0505           SUBB    RH6,5;              % (IM) MODE
4206 4300           SUBB    RH6,LAB;            % (DA) MODE
```

```
0226                     SUBB    RH6,R2^;          % (IR) MODE
4216 4300                SUBB    RH6,LAB(R1);      % (X)  MODE
5216 4300                SUBL    RR6,LAB(R1);      % (X)  MODE
1226                     SUBL    RR6,R2^;          % (IR) MODE
9246                     SUBL    RR6,RR4;          % (R)  MODE
1206 0000 0005           SUBL    RR6,5;            % (IM) MODE
5206 4300                SUBL    RR6,LAB;          % (DA) MODE

AF46                     TCC     ZR,R4;            % (R)  MODE
AE46                     TCCB    ZR,RH4;           % (R)  MODE
4D04 4300                TEST    LAB;              % (DA) MODE
0D24                     TEST    R2^;              % (IR) MODE
4D14 4300                TEST    LAB(R1);          % (X)  MODE
8D44                     TEST    R4;               % (R)  MODE
4C14 4300                TESTB   LAB(R1);          % (X)  MODE
0C24                     TESTB   R2^;              % (IR) MODE
4C04 4300                TESTB   LAB;              % (DA) MODE
8C44                     TESTB   RH4;              % (R)  MODE
5C08 4300                TESTL   LAB;              % (DA) MODE
9C48                     TESTL   RR4;              % (R)  MODE
1C28                     TESTL   R2^;              % (IR) MODE
5C18 4300                TESTL   LAB(R1);          % (X)  MODE
B8B8 0620                TRDB    R11^,R2^,R6;      % (IR) MODE
B8BC 0620                TRDRB   R11^,R2^,R6;      % (IR) MODE
B8B0 0620                TRIB    R11^,R2^,R6;      % (IR) MODE
B8B4 0620                TRIRB   R11^,R2^,R6;      % (IR) MODE
B8BA 0620                TRTDB   R11^,R2^,R6;      % (IR) MODE
B8BE 062E                TRTDRB  R11^,R2^,R6;      % (IR) MODE
B8B2 0620                TRTIB   R11^,R2^,R6;      % (IR) MODE
B8B6 062E                TRTIRB  R11^,R2^,R6;      % (IR) MODE
0D26                     TSET    R2^;              % (IR) MODE
4D16 4300                TSET    LAB(R1);          % (X)  MODE
4D06 4300                TSET    LAB;              % (DA) MODE
8D46                     TSET    R4;               % (R)  MODE
8C46                     TSETB   RH4;              % (R)  MODE
4C06 4300                TSETB   LAB;              % (DA) MODE
0C26                     TSETB   R2^;              % (IR) MODE
4C16 4300                TSETB   LAB(R1);          % (X)  MODE

0907 0005                XOR     R7,5;             % (IM) MODE
0927                     XOR     R7,R2^;           % (IR) MODE
4907 4300                XOR     R7,LAB;           % (DA) MODE
4917 4300                XOR     R7,LAB(R1);       % (X)  MODE
8947                     XOR     R7,R4;            % (R)  MODE
4817 4300                XORB    RH7,LAB(R1);      % (X)  MODE
0827                     XORB    RH7,R2^;          % (IR) MODE
4807 4300                XORB    RH7,LAB;          % (DA) MODE
8847                     XORB    RH7,RH4;          % (R)  MODE
0807 0505                XORB    RH7,5;            % (IM) MODE
```

APPENDIX D

AmZ8000 Instruction Set: Topical Index

Instruction Description	Mnemonic	Data Types	Addressing Modes	Flags Affected
Arithmetic				
Add with Carry	ADC	B, W	R	C, Z, S, V, D[1], H[1]
Add	ADD	B, W, L	R, IM, IR, DA, X	C, Z, S, V, D[1], H[1]
Compare (Immediate)	CP	B, W	IR, DA, X	C, Z, S, V
Compare (Register)	CP	B, W, L	R, IM, IR, DA, X	C, Z, S, V
Decimal Adjust Bit	DAB	B	IR	C, Z, S
Decrement	DEC	B, W	R, IR, DA, X	Z, S, V
Divide	DIV	W, L	R, IM, IR, DA, X	C, Z, S, V
Extend Sign	EXTS	B, W, L	R	C, Z, S, V
Increment	INC	B, W	R, IR, DA, X	Z, S, V
Multiply	MULT	W, L	R, IM, IR, DA, X	C, Z, S, V
Negate	NEG	B, W	R, IR, DA, X	C, Z, S, V
Subtract with Carry	SBC	B, W	R	C, Z, S, V, D[1], H[1]
Subtract	SUB	B, W, L	R, IM, IR, DA, X	C, Z, S, V, D[1], H[1]
Bit Manipulation				
Bit Test	BIT	B, W	R	Z
Bit Reset (Static)	RES	B, W	R, IR, DA, X	–
Bit Reset (Dynamic)	RES	B, W	R	–
Bit Set (Static)	SET	B, W	R, IR, DA, X	–
Bit Set (Dynamic)	SET	B, W	R	–
Bit Test and Set	TSET	B, W	R, IR, DA, X	S
Block Transfer and String Manipulation				
Compare and Decrement	CPD	B, W	IR	C, Z, S, V
Compare, Decrement, and Repeat	CPDR	B, W	IR	C, Z, S, V
Compare and Increment	CPI	B, W	IR	C, Z, S, V
Compare, Increment, and Repeat	CPIR	B, W	IR	C, Z, S, V
Compare String and Decrement	CPSD	B, W	IR	C, Z, S, V
Compare String, Decrement, and Repeat	CPSDR	B, W	IR	C, Z, S, V
Compare String and Increment	CPSI	B, W	IR	C, Z, S, V
Compare String, Increment, and Repeat	CPSIR	B, W	IR	C, Z, S, V
Load and Decrement	LDD	B, W	IR	V
Load, Decrement, and Repeat	LDDR	B, W	IR	V
Load and Increment	LDI	B, W	IR	V
Load, Increment, and Repeat	LDIR	B, W	IR	V
Translate and Decrement	TRDB	B	IR	Z, V
Translate, Decrement, and Repeat	TRDRB	B	IR	Z, V
Translate and Increment	TRIB	B	IR	Z, V
Translate, Increment, and Repeat	TRIRB	B	IR	Z, V
Translate, Test, and Decrement	TRTDB	B	IR	Z, V
Translate, Test, Decrement, Repeat	TRTDRB	B	IR	Z, V
Translate, Test, and Increment	TRTIB	B	IR	Z, V
Translate, Test, Increment, and Repeat	TRTIRB	B	IR	Z, V
CPU Control Instructions				
Complement Flag	COMFLG	–	–	C[2], Z[2], S[2], P[2], V[2]
Disable Interrupt	DI	–	–	–
Enable Interrupt	EI	–	–	–
Halt	HALT	–	–	–
Load Control Register (from register)	LDCTL	–	R	C[2], Z[2], S[2], P[2], D[2], H[2]
Load Control Register (to register)	LDCTL	–	–	–
Load Program Status	LDPS	–	IR, DA, X	C, Z, S, P, D, H
Multi-Bit Test	MBIT	–	–	S
Multi-Micro Request	MREQ	–	–	Z, S
Multi-Micro Reset	MRES	–	–	–
Multi-Micro Set	MSET	–	–	–
No Operation	NOP	–	–	–
Reset Flag	RESFLG	–	–	C[2], Z[2], S[2], P[2], V[2]
Set Flag	SETFLG	–	–	C[2], Z[2], S[2], P[2], V[2]

[1] Flag affected only for byte operation.

[2] Flag modified only if specified by the instruction.

AmZ8000 Instruction Set: Topical Index (Cont.)

Instruction Description	Mnemonic	Data Types	Addressing Modes		Flags Affected
Input/Output Instructions[3]			Regular	Special	
Input	(S)IN[3]	B, W	IR, DA	(DA)	–
Input and Decrement	(S)IND[3]	B, W	IR	(IR)	V
Input, Decrement and Repeat	(S)INDR[3]	B, W	IR	(IR)	V
Input and Increment	(S)INI[3]	B, W	IR	(IR)	V
Input, Increment, and Repeat	(S)INIR[3]	B, W	IR	(IR)	V
Output	(S)OUT[3]	B, W	IR, DA	(DA)	–
Output and Decrement	(S)OUTD[3]	B, W	IR	(IR)	V
Output, Decrement, and Repeat	(S)OUTDR[3]	B, W	IR	(IR)	V
Output and Increment	(S)OUTI[3]	B, W	IR	(IR)	V
Output, Increment, and Repeat	(S)OUTIR[3]	B, W	IR	(IR)	V
Logical Instructions					
And	AND	B, W	R, IM, IR, DA, X		Z, S, P
Complement	COM	B, W	R, IR, DA, X		Z, S, P
Or	OR	B, W	R, IM, IR, DA, X		Z, S, P
Test	TEST	B, W, L	R, IR, DA, X		Z, S, P
Test Condition Code	TCC	B, W	R		–
Exclusive Or	XOR	B, W	R, IM, IR, DA, X		Z, S, P
Program Control Instructions					
Call Procedure	CALL	–	IR, DA, X		–
Call Procedure Relative	CALR	–	RA		–
Decrement, Jump if Not Zero	DJNZ	B, W	RA		
Interrupt Return	IRET	–	–		C, Z, S, P, D, H
Jump	JP	–	IR, DA, X		–
Jump Relative	JR	–	RA		–
Return from Procedure	RET	–	–		–
System Call	SC	–	–		–
Rotate and Shift Instructions					
Rotate Left	RL	B, W	R		–
Rotate Left Through Carry	RLC	B, W	R		C, Z, S, V
Rotate Left Digit	RLDB	B	R		Z, S
Rotate Right	RR	B, W	R		C, Z, S, V
Rotate Right Through Carry	RRC	B, W	R		C, Z, S, V
Rotate Right Digit	RRDB	B	R		Z, S
Shift Dynamic Arithmetic	SDA	B, W, L	R		C, Z, S, V
Shift Dynamic Logical	SDL	B, W, L	R		C, Z, S, V
Shift Left Arithmetic	SLA	B, W, L	R		C, Z, S, V
Shift Left Logical	SLL	B, W, L	R		C, Z, S, V
Shift Right Arithmetic	SRA	B, W, L	R		C, Z, S, V
Shift Right Logical	SRL	B, W, L	R		C, Z, S, V

[3] Each I/O instruction has a special counterpart used to alert other devices that a Special I/O transaction is occurring. The special I/O mnemonic is S + regular mnemonic. Refer to section 4.6 for further details.

APPENDIX E
AmZ8000 Instruction Set
Opcode Map

Upper Nibble (Hex), Upper Instruction Byte	Lower Nibble (Hex), Upper Instruction Byte															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	**ADDB** R←IR R←IM	**ADD** R←IR R←IM	**SUBB** R←IR R←IM	**SUB** R←IR R←IM	**ORB** R←IR R←IM	**OR** R←IR R←IM	**ANDB** R←IR R←IM	**AND** R←IR R←IM	**XORB** R←IR R←IM	**XOR** R←IR R←IM	**CPB** R←IR R←IM	**CP** R←IR R←IM	**See Table 1**	**See Table 1**	**Extend Inst**	**Extend Inst**
1	**CPL** R←IR R←IM	**PUSHL** IR←IR	**SUBL** R←IR R←IM	**PUSH** IR←IR	**LDL** R←IR R←IM	**POPL** IR←IR	**ADDL** R←IR R←IM	**POP** IR←IR	**MULTL** R←IR R←IM	**MULT** R←IR R←IM	**DIVL** R←IR R←IM	**DIV** R←IR R←IM	**See Table 2**	**LDL** IR←R	**JP** PC←IR	**CALL** PC←IR
2	**LDB** R←IR R←IM	**LD** R←IR R←IM	**RESB** IR←IM R←R	**RES** IR←IM R←R	**SETB** IR←IM R←R	**SET** IR←IM R←R	**BITB** IR←IM R←R	**BIT** IR←IM R←R	**INCB** IR←IM	**INC** IR←IM	**DECB** IR←IM	**DEC** IR←IM	**EXB** R←IR	**EX** R←IR	**LDB** IR←R	**LD** IR←R
3	**LDB** R←BA **LDRB** R←RA	**LD** R←BA **LDR** R←RA	**LDB** BA←R **LDRB** RA←R	**LD** BA←R **LDR** RA←R	**LDA** R←BA **LDAR** R←RA	**LDL** R←BA **LDRL** R←RA	**RSVD**	**LDL** BA←R **LDRL** RA←R	**RSVD**	**LDPS** IR	**See Table 3**	**See Table 3**	**INB** R←IR	**IN** R←IR	**OUTB** IR←R	**OUT** IR←R
4	**ADDB** R←X R←DA	**ADD** R←X R←DA	**SUBB** R←X R←DA	**SUB** R←X R←DA	**ORB** R←X R←DA	**OR** R←X R←DA	**ANDB** R←X R←DA	**AND** R←X R←DA	**XORB** R←X R←DA	**XOR** R←X R←DA	**CPB** R←X R←DA	**CP** R←X R←DA	**See Table 1**	**See Table 1**	**Extend Inst**	**Extend Inst**
5	**CPL** R←X R←DA	**PUSHL** IR←X IR←DA	**SUBL** R←X R←DA	**PUSH** IR←X IR←DA	**LDL** R←X R←DA	**POPL** IR←X IR←DA	**ADDL** R←X R←DA	**POP** IR←X IR←DA	**MULTL** R←X R←DA	**MULT** R←X R←DA	**DIVL** R←X R←DA	**DIV** R←X	**See Table** 2	**LDL** X←R DA←R	**JP** PC←X PC←DA	**CALL** PC←X PC←DA
6	**LDB** R←X R←DA	**LD** R←X R←DA	**RESB** X←IM DA←IM	**RES** X←IM DA←IM	**SETB** X←IM DA←IM	**SET** X←IM DA←IM	**BITB** X←IM DA←IM	**BIT** X←IM DA←IM	**INCB** X←IM DA←IM	**INC** X←IM DA←IM	**DECB** X←IM DA←IM	**DEC** X←IM DA←IM	**EXB** R↔X R↔DA	**EX** R↔X R↔DA	**LDB** X↔R DA↔R	**LD** X↔R DA↔R
7	**LDB** R←BX	**See Table 7**	**LDB** BX←R	**LD** BX←R	**LDA** R←BX	**LDL** R←BX	**LDA** R←X R←DA	**LDL** BX←R	**RSVD**	**LDPS** PS←X PS←DA	**HALT**	**See Table 7**	**EI** **DI**	**See Table 7**	**RSVD**	**SC**
8	**ADDB** R←R	**ADD** R←R	**SUBB** R←R	**SUB** R←R	**ORB** R←R	**OR** R←R	**ANDB** R←R	**AND** R←R	**XORB** R←R	**XOR** R←R	**CPB** R←R	**CP** R←R **1**	**See Table 1**	**See Table**	**Extend Inst.**	**Extend Inst.**
9	**CPL** R←R	**PUSHL** IR←R	**SUBL** R←R	**PUSH** IR←R	**LDL** R←R	**POPL** R←IR	**ADDL** R←R	**POP** R←IR	**MULTL** R←R	**MULT** R←R	**DIVL** R←R	**DIV** R←R	**See Table 2**	**RSVD**	**RET** PC←(SP)	**RSVD**
A	**LDB** R←R	**LD** R←R	**RESB** R←IM	**RES** R←IM	**SETB** R←IM	**SET** R←IM	**BITB** R←IM	**BIT** R←IM	**INCB** R←IM	**INC** R←IM	**DECB** R←IM	**DEC** R←IM	**EXB** R↔R	**EX** R↔R	**TCCB** R	**TCC** R
B	**DAB** R	**EXTS** **EXTSB** **EXTSL** **R**	**See Table 4**	**See Table 4**	**ADCB** R←R	**ADC** R←R	**SBCB** R←R	**SBC** R←R	**See Table 5**	**RSVD**	**See Table 6**	**See Table 6**	**RRDB** R	**LDK** R←IM	**RLDB** R	**RSVD**
C	**LDB** R←IM															→
D	**CALR** PC←RA															→
E	**JR** PC←RA															→
F	**DJNZ** **DBJNZ** PC←RA															→

Notes:
1. Reserved Instructions (RSVD) should not be used. The result of their execution is not defined.
2. The execution of an extended instruction will result in an Extended Instruction Trap if the EPE bit in the FCW is a zero. If the flag is a one the Extended Instruction will be executed by the EPU function.

OPCODE MAP (Cont.)

TABLE 4. UPPER INSTRUCTION BYTE

Lower Nibble (Hex), Lower Instruction Byte	B2	B3
0	**RLB** (1 bit) R	**RL** (1 bit) R
1	**SLLB** R **SRLB** R	**SLL** R **SRL** R
2	**RLB** (2 bits) R	**RL** (2 bits) R
3	**SDLB** R	**SDL** R
4	**RRB** (1 bit) R	**RR** (1 bit) R
5	RSVD	**SLLL** R **SRLL**
6	**RRB** (2 bits) R	**RR** (2 bits) R
7	RSVD	**SDLL** R
8	**RLCB** (1 bit) R	**RLC** (1 bit) R
9	**SLAB** R **SRAB** R	**SLA** R **SRA** R
A	**RLCB** (2 bits) R	**RLC** (2 bits) R
B	**SDAB** R	**SDA** R
C	**RRCB** (1 bit) R	**RRC** (1 bit) R
D	RSVD	**SLAL** R **SRAL**
E	**RRCB** (2 bits) R	**RRC** (2 bits) R
F	RSVD	**SDAL** R

TABLE 5.

Lower Nibble (Hex), Lower Instruction Byte	B8
0	**TRIB** IR
1	RSVD
2	**TRTIB** IR
3	RSVD
4	**TRIRB** IR
5	RSVD
6	**TRTIRB** IR
7	RSVD
8	**TRDB** IR
9	RSVD
A	**TRTDB** IR
B	RSVD
C	**TRDRB** IR
D	RSVD
E	**TRTDRB** IR
F	RSVD

OPCODE MAP (Cont.)

TABLE 1. UPPER INSTRUCTION BYTE

Lower Nibble (Hex), Lower Instruction Byte	OC	OD	4C	4D	8C	8D
0	**COMB** IR	**COM** IR	**COMB** X DA	**COM** X DA	**COMB** R	**COM** R
1	**CPB** IR, IM	**CP** IR, IM	**CPB** X, IM DA, IM	**CP** X, IM DA, IM	**LCTLB** R←FLGS	**SETFLG**
2	**NEGB** IR	**NEG** IR	**NEGB** X DA	**NEG** X DA	**NEGB** R	**NEG** R
3	RSVD	RSVD	RSVD	RSVD	RSVD	**RESFLG**
4	**TESTB** IR	**TEST** IR	**TESTB** X DA	**TEST** X DA	**TESTB** R	**TEST** R
5	**LDB** IR←IM	**LD** IR←IM	**LDB** X←IM DA←IM	**LD** X←IM DA←IM	RSVD	**COMFLG**
6	**TSETB** IR	**TSET** IR	**TSETB** X DA	**TSET** X DA	**TSETB** R	**TSET** R
7	RSVD	RSVD	RSVD	RSVD	RSVD	**NOP**
8	**CLRB** IR	**CLR** IR	**CLRB** X DA	**CLR** X DA	**CLRB** R	**CLR** R
9		**PUSH** IM			**LDCTLB** FLGS←R	

TABLE 2. UPPER INSTRUCTION BYTE

Lower Nibble (Hex) Lower Instruction Byte	1C	5C	9C
0	RSVD	RSVD	RSVD

1	**LDM**
	R←IR
		LDM
		R←X
		R←DA	

8	**TESTL** IR	**TESTL** X DA	**TESTL** R

9	**LDM**
	IR←R
		LDM
		X←R
		DA←R	

TABLE 3. UPPER INSTRUCTION BYTE

Lower Nibble (Hex), Lower Instruction Byte	3A	3B
0	**INIB** IR←IR **INIRB** IR←IR	**INI** IR←IR **INIR** IR←IR
1	**SINIB** IR←IR **SINIRB** IR←IR	**SINI** IR←IR **SINIR** IR←IR
2	**OUTIB** IR←IR **OTIRB** IR←IR	**OUTI** IR←IR **OUTIR** IR←IR
3	**SOUTIB** IR←IR **SOTIRB** IR←IR	**SOUTI** IR←IR **SOTIR** IR←IR
4	**INB** R←DA	**IN** R←DA
5	**SINB** R←DA	**SIN** R←DA
6	**OUTB** DA←R	**OUT** DA←R
7	**SOUTB** DA←R	**SOUT** DA←R
8	**INDB** IR←IR **INDRB** IR←IR	**IND** IR←IR **INDR** IR←IR
9	**SINDB** IR←IR **SINDRB** IR←IR	**SIND** IR←IR **SINDR** IR←IR
A	**OUTDB** IR←IR **OTDRB** IR←IR	**OUTD** IR←IR **OTDR** IR←IR
B	**SOUTDB** IR←IR **SOTDRB** IR←IR	**SOUTD** IR←IR **SOTDR** IR←IR

OPCODE MAP (Cont.)

TABLE 6.

Lower Nibble (Hex), Lower Instruction Byte	BA	BB
0	CPIB IR	CPI IR
1	LDIB IR←IR LDIRB IR←IR	LDI IR←IR LDIR IR←IR
2	CPSIB IR	CPSI IR
3	RSVD	RSVD
4	CPRIB IR	CPIR IR
5	RSVD	RSVD
6	CPSIRB IR	CPSIR IR
7	RSVD	RSVD
8	CPDB IR	CPD IR
9	LDDB IR←IR LDDRB IR←IR	LDD IR←IR LDDR IR←IR
A	CPSDB IR	CPSD IR
B	RSVD	RSVD
C	CPDRB IR	CPDR IR
D	RSVD	RSVD
E	CPSDRB IR	CPSDR IR
F	RSVD	RSVD

TABLE 7.

Lower Nibble (Hex), Lower Instruction Byte	7B	7D
0	IRET PC←(SSP)	RSVD
1	RSVD	RSVD
2	RSVD	LDCTL R←FCW
3	RSVD	LDCTL R←RFRSH
4	RSVD	LDCTL R← PSAPSEG
5	RSVD	LDCTL R← PSAPOFF
6	RSVD	LDCTL R←NSPSEG
7	RSVD	LDCTL R←NSPOFF
8	MSET	RSVD
9	MRES	RSVD
A	MBIT	LDCTL FCW←R
B	RSVD ↓	LDCTL RFRSH←R
C	↓	LDCTL PSAPSEG ←R
D	MREQ R	LDCTL PSAPOFF ←R
E	RSVD	LDCTL NSPSEG←R
F	RSVD	LDCTL NSPOFF←R

APPENDIX F
EXECUTIVE MODULE SAMPLE CODE

The following code is taken from the Executive Module of MONITOR3, a sample program developed at the AMD Customer Education Center by Charles R. McCallan and Bruce W. Pettner. MONITOR3 was developed to demonstrate the entire AmZ8002 instruction set, MACRO8000 (MACZ) and the linker.

MONITOR3 is the principal vehicle of instruction used in the center's course ED8000B, Assembly Language Programming for the AmZ8000. A complete listing of MONITOR3 is available to students enrolling in the ED8000S or ED8000B seminars. It is reprinted in the ED8000A/B STUDY GUIDE.

MONITOR3

A SIMPLE MONITOR

The following listings are for a simple monitor program for the AmZ8002. The program is written in 15 modules and linked together. The listings include the linking operation, which was done from a directives file.

This 'simple-minded' monitor was written to demonstrate the **MACZ** (MACRO8000) macro assembler and **LNKZ** (LINK8000) linker which support the AmZ8000. The code is an attempt to demonstrate various methods of programming the AmZ8000 with a consious effort to do things in as many ways as possible while maintaining good programming practices. The main intent of the program is to show example AmZ8000 code that is structured and WELL DOCUMENTED.

This monitor is NOT intended to be a sophisticated program for end users, but with suitable extentions might form the basis for a useful small monitor.

The 'simple' monitor as implemented supports the following basic commands and DEBUG functions:

ALTER OR DISPLAY MEMORY	(A command)
SET SOFTWARE BREAKPOINT	(B command)
DUMP MEMORY	(D command)
FCW ALTER/DISPLAY	(F command)
GOTO OR RESUME EXECUTION	(G command)
HELP (LIST COMMAND SUMMARY)	(H command)
PROGRAM DOWNLOAD (.BIN FILE)	(L command - dummy module)
MEMORY FILL	(M command)
DISLAY CURRENT PC	(P command)
REGISTER ALTER/DISPLAY	(R command)
DISPLAY SYSTEM STACK POINTER	(S command)
EXIT ANY COMMAND	(ESC)

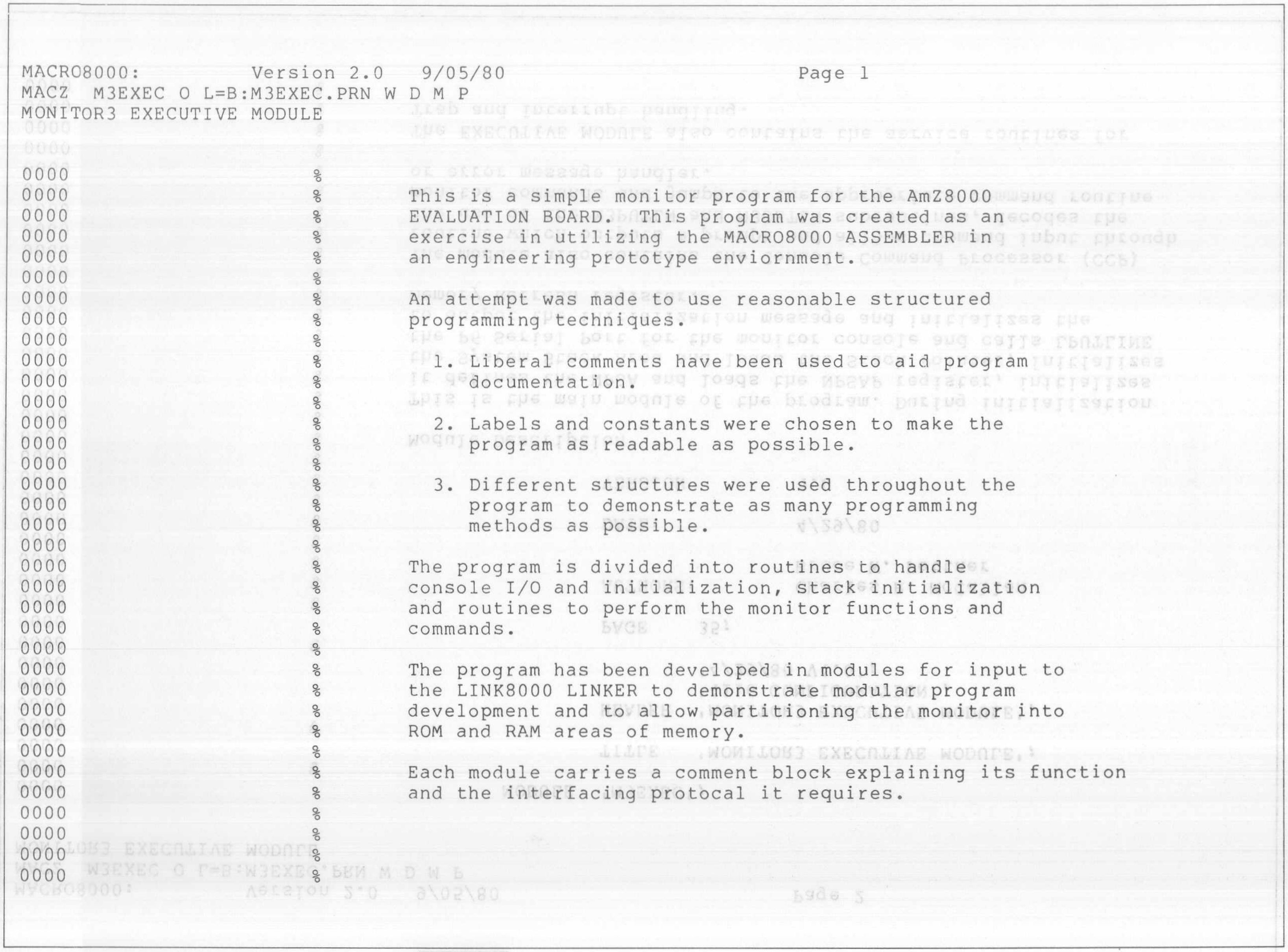
All commands are a single letter. All command arguments are either DECIMAL designated by nn or HEX designated by hhhh (i.e. G4AF0 starts execution at address 4AF0). Alter/display commands always alter a full word (4 HEX digits). To exit ANY command at any time type an esape (ESC key). These conventions greatly simplify the program logic and help keep the code simple and understandable.

If it is desired to add a command the procedure is very simple. The command (single letter) needs to be added to the command table in the 'DATA' module and the entry point label must be entered in the 'ONGOTO' statement in the 'EXEC' modules command decode routine. The entries (command and label) must be in the same relative positions in both the command table and entry (label) list. The new command will then be decoded by the CCP routines and control transferred to your entry point.

Every effort has been made to comment and structure the code so that it is self-documenting. Comment blocks explain each routine and should make following the code very easy.

THIS IS A SAMPLE PROGRAM ONLY AND NO RESPONSIBILITY IS ASSUMED BY ADVANCED MICRO DEVICES OR ADVANCED MICRO COMPUTERS FOR ITS USE.

The monitor as linked will run on the AmZ8000 Evaluation Board but could also be linked for prom generation by setting the absolute assignments of the 'DATA' and 'NPSA' segments to '0' and '256' (decimal).

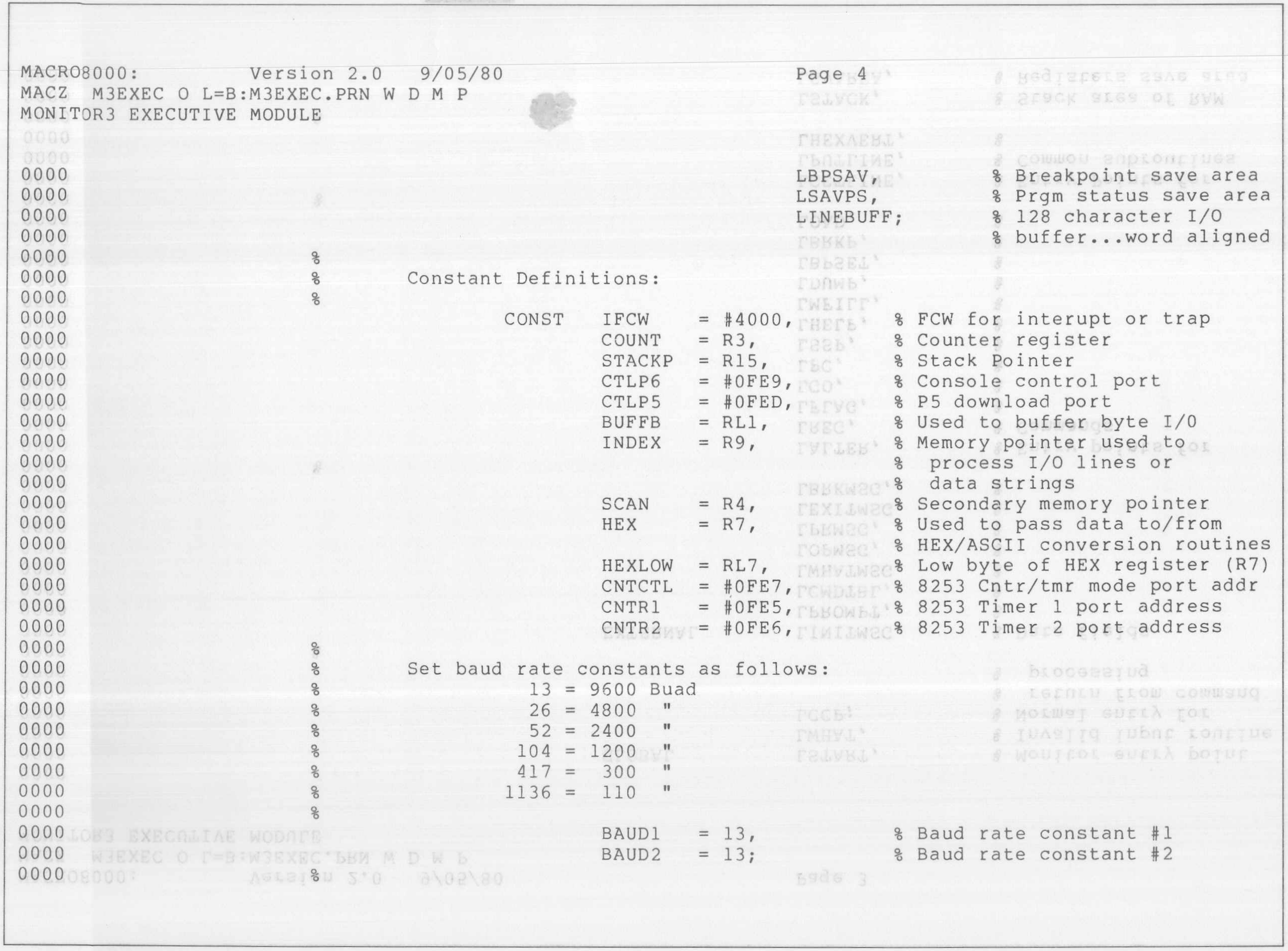


```
MACRO8000:         Version 2.0   9/05/80               Page 1
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0000              %
0000              %    This is a simple monitor program for the AmZ8000
0000              %    EVALUATION BOARD.  This program was created as an
0000              %    exercise in utilizing the MACRO8000 ASSEMBLER in
0000              %    an engineering prototype enviornment.
0000              %
0000              %    An attempt was made to use reasonable structured
0000              %    programming techniques.
0000              %
0000              %     1. Liberal comments have been used to aid program
0000              %      documentation.
0000              %
0000              %     2. Labels and constants were chosen to make the
0000              %        program as readable as possible.
0000              %
0000              %     3. Different structures were used throughout the
0000              %        program to demonstrate as many programming
0000              %        methods as possible.
0000              %
0000              %    The program is divided into routines to handle
0000              %    console I/O and initialization, stack initialization
0000              %    and routines to perform the monitor functions and
0000              %    commands.
0000              %
0000              %    The program has been developed in modules for input to
0000              %    the LINK8000 LINKER to demonstrate modular program
0000              %    development and to allow partitioning the monitor into
0000              %    ROM and RAM areas of memory.
0000              %
0000              %    Each module carries a comment block explaining its function
0000              %    and the interfacing protocal it requires.
0000              %
0000              %
0000              %
0000              %
```

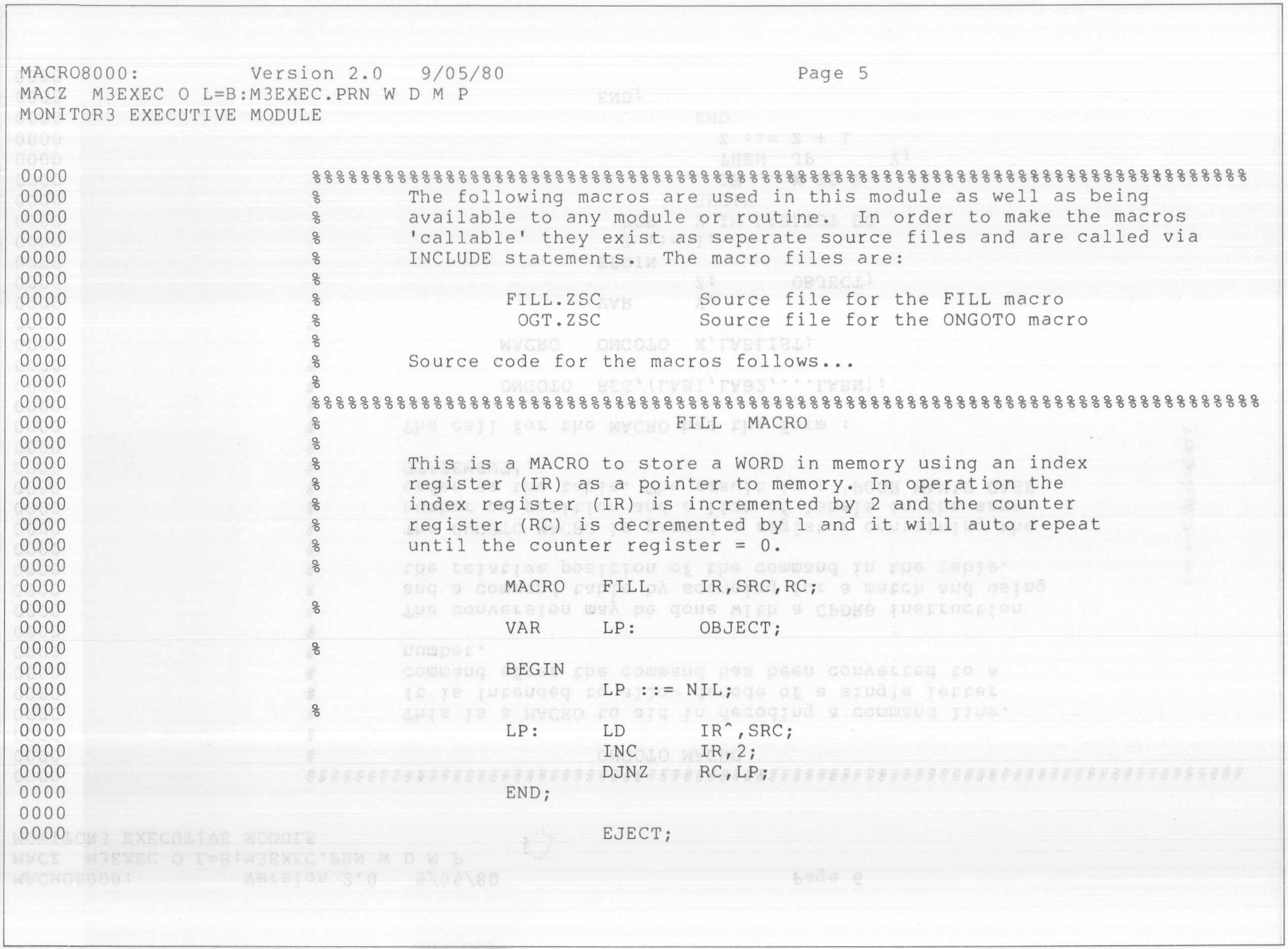
A

```
MACRO8000:         Version 2.0   9/05/80                         Page 2
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

0000                                    MODULE  'M3EXEC';
0000                    %
0000                                            TITLE   'MONITOR3 EXECUTIVE MODULE';
0000                    %
0000                                            HEADER 'MONITOR3 EXECUTIVE MODULE',
0000                                                   '4116 CONFIGURATION',
0000                                                   '4/29/80 V4.0';
0000                    %
0000                                            PAGE    35;
0000                    %
0000                    %                       AUTHORS:        Charles R. McCallan
0000                    %                                       Bruce W. Pettner
0000                    %
0000                    %                       DATE:           4/29/80
0000                    %
0000                    %                       VERSION         4.0
0000                    %
0000                    %       Module Description
0000                    %
0000                    %       This is the main module of the program. During initialization
0000                    %       it defines the NPSA and loads the NPSAP register, initializes
0000                    %       the System Stack Area and loads the Stack Pointer, initializes
0000                    %       the P6 Serial Port for the monitor console and calls LPUTLINE
0000                    %       to output the initialization message and initializes the
0000                    %       memory Refresh register.
0000                    %
0000                    %       The module also contains the Console Command Processor (CCP)
0000                    %       routine which outputs a prompt and allows command input through
0000                    %       the use of the M3PUTLN and M3GETLN subroutines, decodes the
0000                    %       monitor commands and jumps to the appropriate command routine
0000                    %       or error message handler.
0000                    %
0000                    %       The EXECUTIVE MODULE also contains the service routines for
0000                    %       Trap and Interrupt handling.
0000                    %
```

```
MACRO8000:          Version 2.0   9/05/80                         Page 3
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0000
0000                                              GLOBAL          LSTART,          % Monitor entry point
0000                                                              LWHAT,           % Invalid input routine
0000                                                              LCCP;            % Normal entry for
0000                                                                               %  return from command
0000                                                                               %  processing
0000                     %
0000                                              EXTERNAL        LINITMSG,        % Data fields
0000                                                              LPROMPT,         %
0000                                                              LCMDTBL,         %
0000                                                              LWHATMSG,        %
0000                                                              LOPMSG,          %
0000                                                              LPRMSG,          %
0000                                                              LEXITMSG,        %
0000                                                              LBRKMSG,         %
0000                     %
0000                                                              LALTER,          % Entry points for
0000                                                              LREG,            % Commands
0000                                                              LFLAG,           %
0000                                                              LGO,             %
0000                                                              LPC,             %
0000                                                              LSSP,            %
0000                                                              LHELP,           %
0000                                                              LMFILL,          %
0000                                                              LDUMP,           %
0000                                                              LBPSET,          %
0000                                                              LBRKP,           %
0000                                                              LOAD,            %
0000                     %
0000                                                              LGETLINE,        % Entry Points for
0000                                                              LPUTLINE,        % Common subroutines
0000                                                              LHEXVERT,        %
0000                     %
0000                                                              LSTACK,          % Stack area of RAM
0000                                                              LSVAREA,         % Registers save area
```



```
MACRO8000:         Version 2.0   9/05/80                        Page 4
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0000                                                            LBPSAV,         % Breakpoint save area
0000                                                            LSAVPS,         % Prgm status save area
0000                                                            LINEBUFF;       % 128 character I/O
0000                                                                            % buffer...word aligned
0000                    %
0000                    %       Constant Definitions:
0000                    %
0000                                    CONST   IFCW    = #4000,        % FCW for interupt or trap
0000                                            COUNT   = R3,           % Counter register
0000                                            STACKP  = R15,          % Stack Pointer
0000                                            CTLP6   = #0FE9,        % Console control port
0000                                            CTLP5   = #0FED,        % P5 download port
0000                                            BUFFB   = RL1,          % Used to buffer byte I/O
0000                                            INDEX   = R9,           % Memory pointer used to
0000                                                                    %  process I/O lines or
0000                                                                    %  data strings
0000                                            SCAN    = R4,           % Secondary memory pointer
0000                                            HEX     = R7,           % Used to pass data to/from
0000                                                                    % HEX/ASCII conversion routines
0000                                            HEXLOW  = RL7,          % Low byte of HEX register (R7)
0000                                            CNTCTL  = #0FE7,        % 8253 Cntr/tmr mode port addr
0000                                            CNTR1   = #0FE5,        % 8253 Timer 1 port address
0000                                            CNTR2   = #0FE6,        % 8253 Timer 2 port address
0000                    %
0000                    %       Set baud rate constants as follows:
0000                    %                 13 = 9600 Buad
0000                    %                 26 = 4800  "
0000                    %                 52 = 2400  "
0000                    %                104 = 1200  "
0000                    %                417 =  300  "
0000                    %               1136 =  110  "
0000                    %
0000                                            BAUD1   = 13,           % Baud rate constant #1
0000                                            BAUD2   = 13;           % Baud rate constant #2
0000                    %
```



```
MACRO8000:         Version 2.0   9/05/80                        Page 5
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE
0000                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0000                    %       The following macros are used in this module as well as being
0000                    %       available to any module or routine. In order to make the macros
0000                    %       'callable' they exist as seperate source files and are called via
0000                    %       INCLUDE statements. The macro files are:
0000                    %
0000                    %               FILL.ZSC        Source file for the FILL macro
0000                    %                OGT.ZSC        Source file for the ONGOTO macro
0000                    %
0000                    %       Source code for the macros follows...
0000                    %
0000                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0000                    %                             FILL MACRO
0000                    %
0000                    %       This is a MACRO to store a WORD in memory using an index
0000                    %       register (IR) as a pointer to memory. In operation the
0000                    %       index register (IR) is incremented by 2 and the counter
0000                    %       register (RC) is decremented by 1 and it will auto repeat
0000                    %       until the counter register = 0.
0000                    %
0000                            MACRO   FILL    IR,SRC,RC;
0000                    %
0000                            VAR     LP:     OBJECT;
0000                    %
0000                            BEGIN
0000                                    LP ::= NIL;
0000                    %
0000                            LP:     LD      IR^,SRC;
0000                                    INC     IR,2;
0000                                    DJNZ    RC,LP;
0000                            END;
0000
0000                                    EJECT;
```

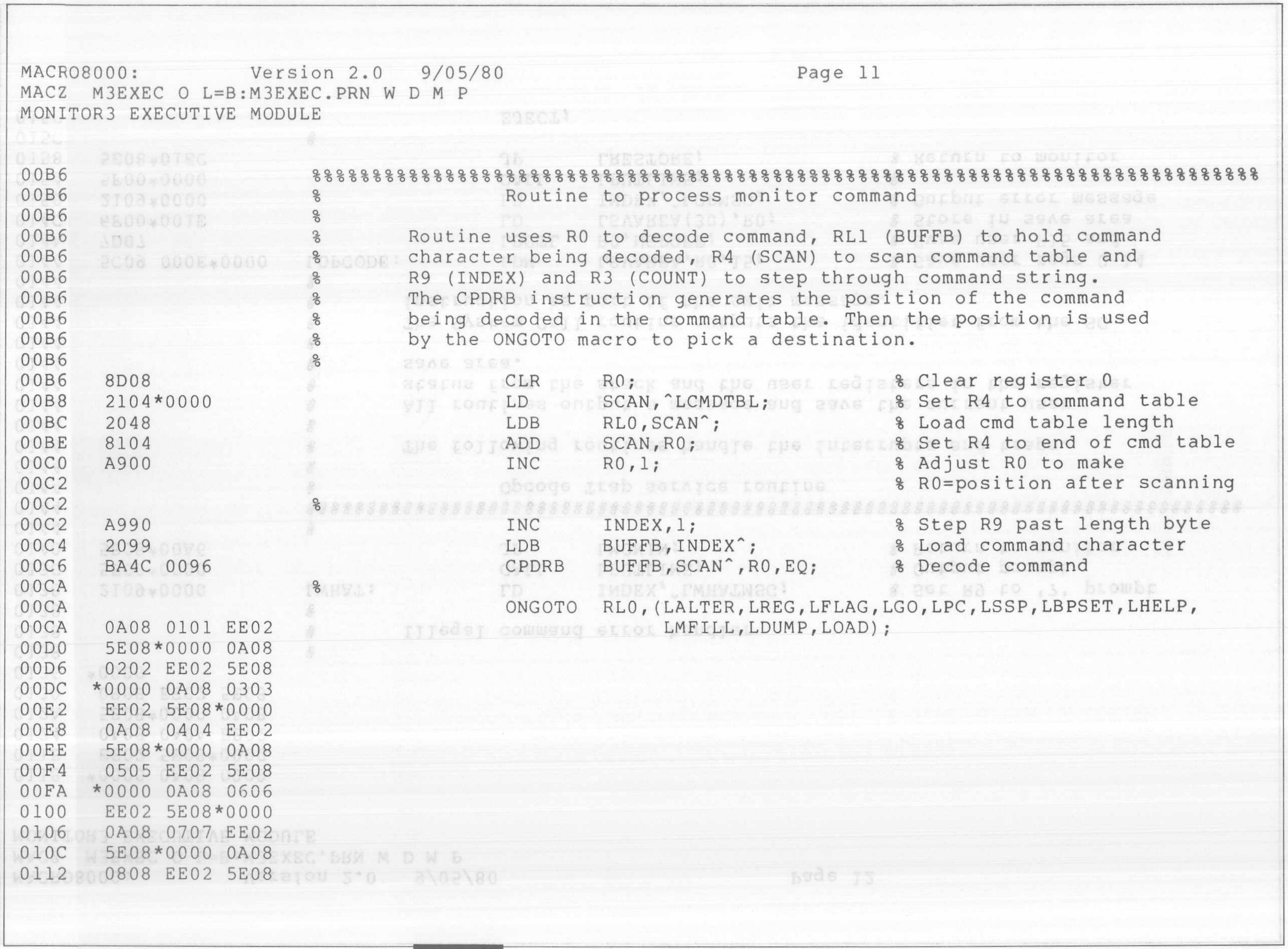
```
MACRO8000:         Version 2.0   9/05/80                        Page 6
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

0000                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0000                    %                       ONGOTO MACRO
0000                    %
0000                    %       This is a MACRO to aid in decoding a command line.
0000                    %       It is intended to allow decode of a single letter
0000                    %       command after the command has been converted to a
0000                    %       number.
0000                    %
0000                    %       The conversion may be done with a CPDRB instruction
0000                    %       and a command table by scanning for a match and using
0000                    %       the relative position of the command in the table.
0000                    %
0000                    %       The ONGOTO MACRO is passed a register containing the
0000                    %       number or position and a list of labels in the same
0000                    %       order as the table. The result is a 'POOR MAN'S CASE
0000                    %       STATEMENT'.
0000                    %
0000                    %       The call for the MACRO has the form :
0000                    %
0000                    %               ONGOTO  REG,(LAB1,LAB2,...LABN);
0000                    %
0000                                    MACRO   ONGOTO  X,LABLIST;
0000                    %
0000                                            VAR     Y,
0000                                                    Z:      OBJECT;
0000                                            BEGIN
0000                                              Z ::= 1;
0000                                              FOR   Y IN LABLIST DO
0000                                                    BEGIN
0000                                                      IF    X EQ Z
0000                                                      THEN  JP      Y;
0000                                                      Z ::= Z + 1
0000                                                    END
0000                                            END;
0000
```

```
MACRO8000:          Version 2.0   9/05/80                    Page 7
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

0000                                   SEGMENT 'NPSA';
0000                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0000                 %      This segment is the actual New Program Status Area and MUST be
0000                 %      placed at a Modulo 256 location at link time. If generating a
0000                 %      .BIN file for testing place it at #4400.  If generating a HEX
0000                 %      file for PROMS place it at #0100.
0000                 %
0000                 %
0000  0000 0000      LNPSA:       LONG:   0;                    % Clear unused words
0004                 %
0004   4000                       WORD:   IFCW;                 %
0006  *0144                       WORD:   ^LOPCODE;             % Set up Opcode Trap
0008                 %
0008   4000                       WORD:   IFCW;                 % Set up Privilaged
000A  *015C                       WORD:   ^LPRIV;               % Opcode Trap
000C                 %
000C   4000                       WORD:   IFCW;                 %
000E  *0174                       WORD:   ^LSYSCALL;            % Set up System Call Trap
0010                 %
0010  0000 0000                   LONG:   0;                    % Clear unused (STRAP) words
0014                 %
0014   4000                       WORD:   IFCW;                 % Set up Break Switch
0016  *01D4                       WORD:   ^LBREAK;              % Interrupt (NMI)
0018                 %
0018                              FOR     4 DO
0018  0000 0000 0000                      WORD:   0;            % Clear remainder of NPSA
001E  0000
0020                 %
0020                                   EJECT;
```

```
MACRO8000:          Version 2.0   9/05/80                         Page 8
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE


0020                                         SEGMENT 'CODE';
0000                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0000                  %       *** This is the Main Entry Point of the monitor program ***
0000                  %
0000   2100 BC00      LSTART:          LD      R0,#8000+30*512;          % 30us Refresh rate
0004   7D0B                            LDCTL   REFRESH,R0;               % Load counter
0006   2100*0000                       LD      R0,^LNPSA;                %
000A   7D0D                            LDCTL   PSAPOFF,R0;               % Set up the NPSAP
000C                  %
000C                  %       This routine initializes the system stack
000C                  %
000C   2109*0000                       LD      INDEX,^LSTACK;            % Set index reg to stack area
0010   2103 0080                       LD      COUNT,128;                % Load length in count reg
0014   0D95 0000 A991                  FILL    INDEX,0,COUNT;            % Init Stack and Save areas
001A   F384
001C   210F*0038                       LD      STACKP,^LSTACK + 56;      % Initialize Stack Pointer
0020   4D05*0000 4000                  LD      LSAVPS,IFCW;              % Load default FCW
0026   2100*0000                       LD      R0,^LSTART;               %
002A   6F00*0002                       LD      LSAVPS(2),R0;             % Load default PC
002E                  %
002E                  %       Set Timers for baud rates at P6 and P5 serial ports
002E                  %
002E   C876                            LDB     RL0,#76;                  % Counter 1 mode 3 (P6)
0030   3A86 0FE7                       OUTB    CNTCTL,RL0;               % Send mode command
0034   C8B6                            LDB     RL0,#B6;                  % Counter 2 mode 3 (P5)
0036   3A86 0FE7                       OUTB    CNTCTL,RL0;               % Send mode command
003A   2100 000D                       LD      R0,BAUD1;                 % P6 baud rate
003E   3A86 0FE5                       OUTB    CNTR1,RL0;                % Send LS byte of rate
0042   3A06 0FE5                       OUTB    CNTR1,RH0;                % Send MS byte of rate
0046   2100 000D                       LD      R0,BAUD2;                 % P5 baud rate
004A   3A06 0FE6                       OUTB    CNTR2,RH0;                % Send LS byte of rate
004E   3A86 0FE6                       OUTB    CNTR2,RL0;                % Send MS byte of rate
0052                                   EJECT;
```

```
0052                    %
0052                    %       This routine initializes the 9551 serial ports at P5 and P6
0052                    %
0052                    %       The delay loop is for timing problems encountered when
0052                    %       resetting the serial ports.
0052                    %
0052   2100 FFFF                        LD      R0,#FFFF;                %
0056   F081             LDELAY:         DJNZ    R0,LDELAY;               %
0058                    %
0058   C900                             LDB     BUFFB,0;                 %
005A   3A96 0FE9                        OUTB    CTLP6,BUFFB;             %
005E   3A96 0FED                        OUTB    CTLP5,BUFFB;             %
0062                    %
0062   3A96 0FE9                        OUTB    CTLP6,BUFFB;             % Output 3 nulls to reset
0066   3A96 0FED                        OUTB    CTLP5,BUFFB;             %
006A                    %
006A   3A96 0FE9                        OUTB    CTLP6,BUFFB;             %
006E   3A96 0FED                        OUTB    CTLP5,BUFFB;             %
0072                    %
0072   C940                             LDB     BUFFB,#40;               %
0074   3A96 0FE9                        OUTB    CTLP6,BUFFB;             % Output reset command
0078   3A96 0FED                        OUTB    CTLP5,BUFFB;             %
007C                    %
007C   C9CE                             LDB     BUFFB,#CE;               % Set modes
007E   3A96 0FE9                        OUTB    CTLP6,BUFFB;             % P6= ASYNC,16X,NO PARITY
0082   C9DE                             LDB     BUFFB,#DE;               % P5= ASYNC,16X,ODD PARITY
0084   3A96 0FED                        OUTB    CTLP5,BUFFB;             %
0088   C927                             LDB     BUFFB,#27;               %
008A   3A96 0FE9                        OUTB    CTLP6,BUFFB;             % Output control commands
008E   3A96 0FED                        OUTB    CTLP5,BUFFB;             % RTS= ACTIVE,RxE= ENABLED
0092                                                                     % DTR= ACTIVE,TxE= ENABLED
0092                    %
0092                                    EJECT;
```

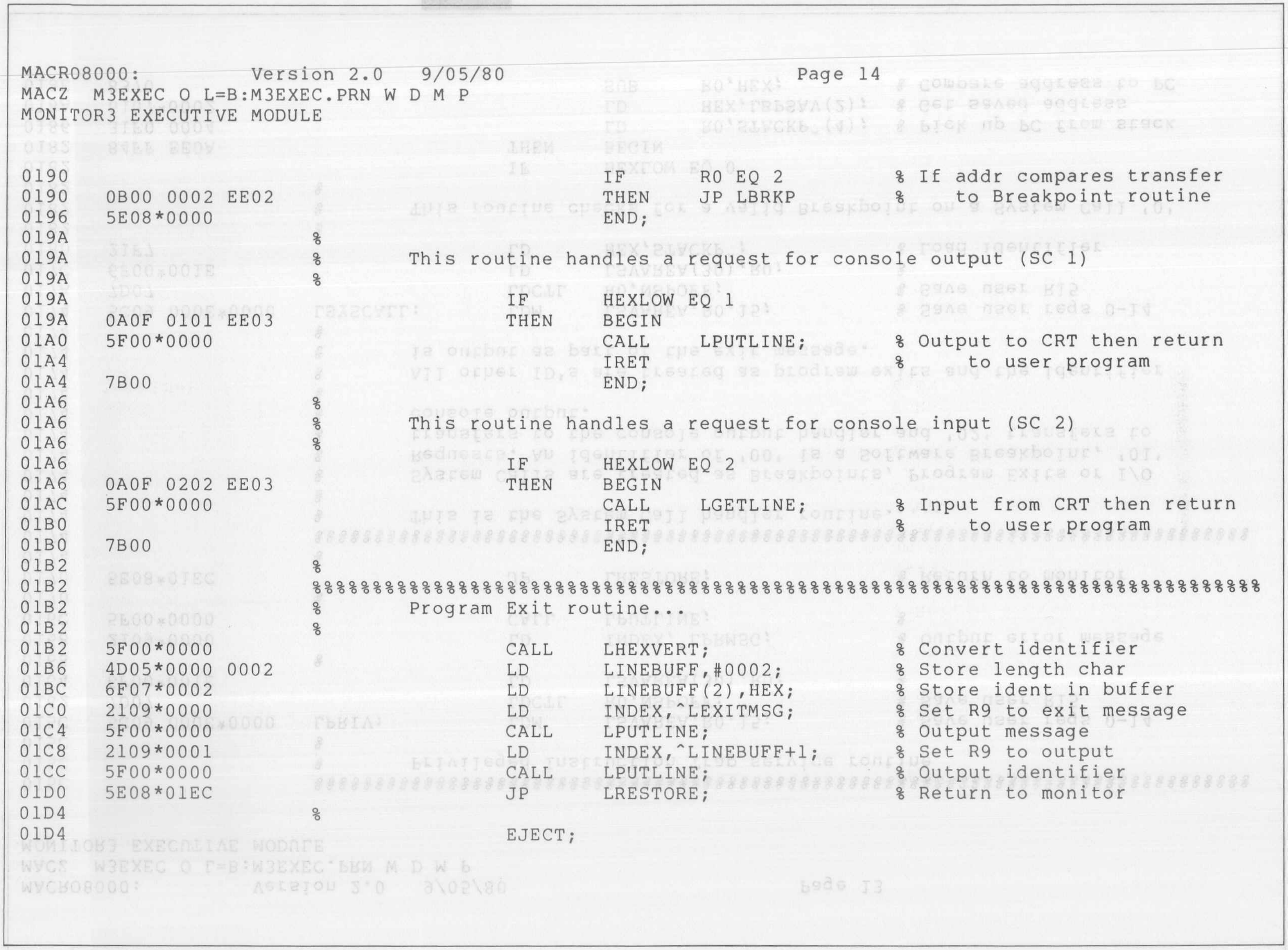
```
MACR08000:          Version 2.0   9/05/80                          Page 10
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

0092                    %
0092                    %       This routine outputs the initialization message
0092                    %
0092    2109*0000                        LD      INDEX,^LINITMSG;         % Set index to message
0096    5F00*0000                        CALL    LPUTLINE;                % Output message
009A    5E08*0000                        JP      LHELP;                   % Go to Help routine to
009E                                                                      %   output instructions
009E                    %
009E                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
009E                    %       This is the Console Command Procrssor routine (CCP)
009E                    %
009E                    %       Thie routine functions as the Executive routine for the
009E                    %       Montior3 program.
009E                    %
009E                    %       CCP controls prompting the user, input and decoding of monitor
009E                    %       commands and calls the routines that execute the monitor commands
009E                    %
009E                    %
009E                    %       Routine to output monitor prompt
009E                    %
009E    2109*0000       LCCP:            LD      INDEX,^LPROMPT;          % Set index to prompt
00A2    5F00*0000                        CALL    LPUTLINE;                % Output prompt
00A6                    %
00A6                    %       Routine to input monitor command
00A6                    %
00A6    2109*0000       LMONIN:          LD      INDEX,^LINEBUFF;         % Set index to buffer
00AA    0C95 4F4F                        LDB     INDEX^,79;               % Set input length
00AE    5F00*0000                        CALL    LGETLINE;                % Input command
00B2    5E04*009E                        JP      OV,LCCP;                 % Retry if ESC was input
00B6                    %
00B6                                     EJECT;
```



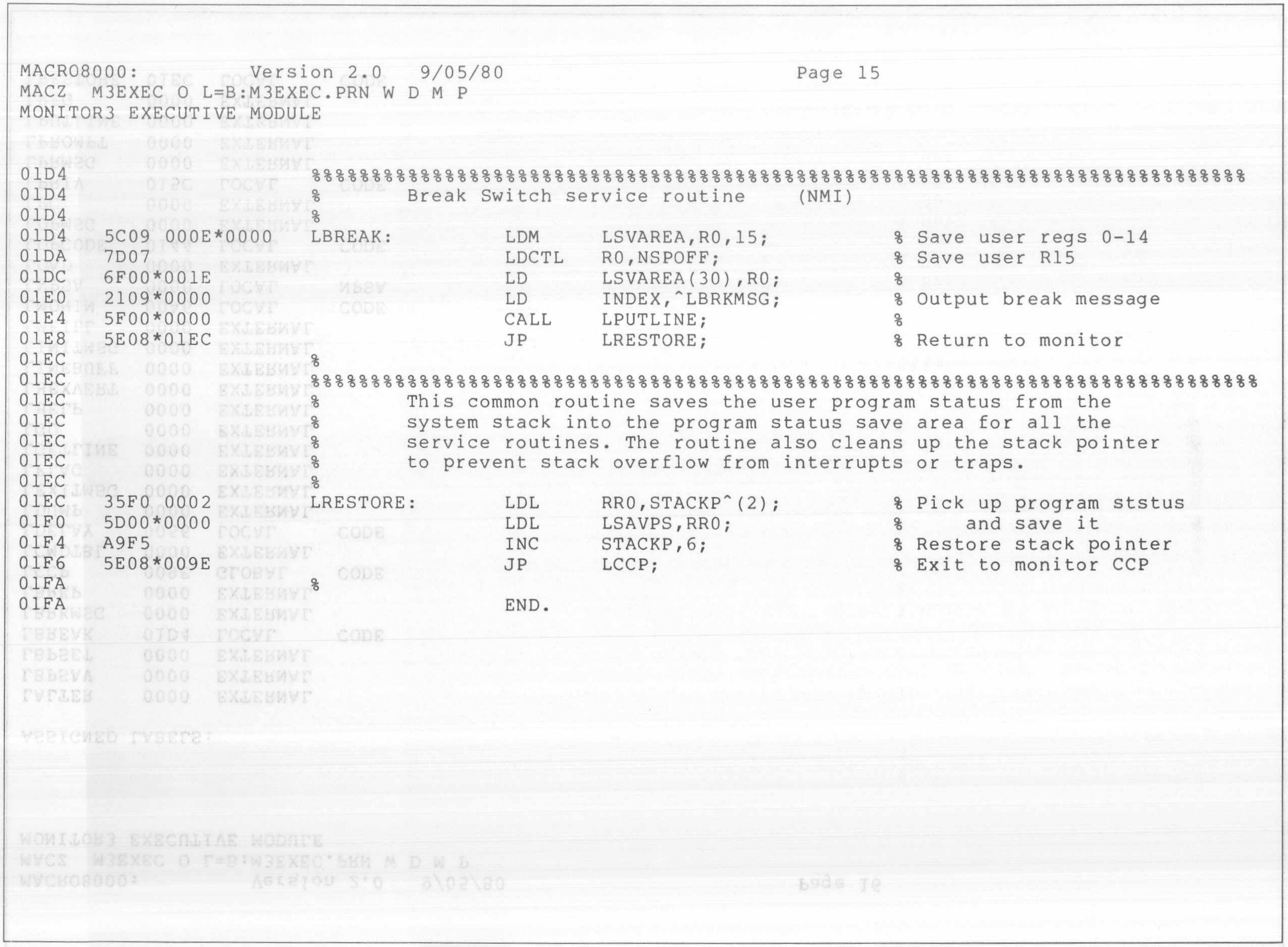
```
MACRO8000:          Version 2.0   9/05/80                     Page 11
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

00B6                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
00B6                    %               Routine to process monitor command
00B6                    %
00B6                    %       Routine uses R0 to decode command, RL1 (BUFFB) to hold command
00B6                    %       character being decoded, R4 (SCAN) to scan command table and
00B6                    %       R9 (INDEX) and R3 (COUNT) to step through command string.
00B6                    %       The CPDRB instruction generates the position of the command
00B6                    %       being decoded in the command table. Then the position is used
00B6                    %       by the ONGOTO macro to pick a destination.
00B6                    %
00B6   8D08                             CLR     R0;                     % Clear register 0
00B8   2104*0000                        LD      SCAN,^LCMDTBL;          % Set R4 to command table
00BC   2048                             LDB     RL0,SCAN^;              % Load cmd table length
00BE   8104                             ADD     SCAN,R0;                % Set R4 to end of cmd table
00C0   A900                             INC     R0,1;                   % Adjust R0 to make
00C2                                                                    % R0=position after scanning
00C2                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
00C2   A990                             INC     INDEX,1;                % Step R9 past length byte
00C4   2099                             LDB     BUFFB,INDEX^;           % Load command character
00C6   BA4C 0096                        CPDRB   BUFFB,SCAN^,R0,EQ;      % Decode command
00CA
00CA                                    ONGOTO  RL0,(LALTER,LREG,LFLAG,LGO,LPC,LSSP,LBPSET,LHELP,
00CA   0A08 0101 EE02                                LMFILL,LDUMP,LOAD);
00D0   5E08*0000 0A08
00D6   0202 EE02 5E08
00DC  *0000 0A08 0303
00E2   EE02 5E08*0000
00E8   0A08 0404 EE02
00EE   5E08*0000 0A08
00F4   0505 EE02 5E08
00FA  *0000 0A08 0606
0100   EE02 5E08*0000
0106   0A08 0707 EE02
010C   5E08*0000 0A08
0112   0808 EE02 5E08
```

```
MACRO8000:         Version 2.0   9/05/80                      Page 12
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

0118  *0000 0A08 0909
011E   EE02 5E08*0000
0124   0A08 0A0A EE02
012A   5E08*0000 0A08
0130   0B0B EE02 5E08
0136  *0000
0138                    %
0138                    %       Illegal command error handler
0138                    %
0138   2109*0000        LWHAT:          LD      INDEX,^LWHATMSG;        % Set R9 to '?' prompt
013C   5F00*0000                        CALL    LPUTLINE;               % Output prompt
0140   5E08*00A6                        JP      LMONIN;                 % Return to monitor
0144                    %
0144                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0144                    %               Opcode Trap service routine
0144                    %
0144                    %       The following routines handle the interrupts and traps.
0144                    %
0144                    %       All routines output a message and save the current user
0144                    %       status from the stack and the user registers in the register
0144                    %       save area.
0144                    %
0144                    %       The System Call routine outputs the identifier from the SC
0144                    %       instruction as part of the exit message.
0144                    %
0144   5C09 000E*0000   LOPCODE:        LDM     LSVAREA,R0,15;          % Save user regs 0-14
014A   7D07                             LDCTL   R0,NSPOFF;              % Svae user R15 and
014C   6F00*001E                        LD      LSVAREA(30),R0;         % Store in save area
0150   2109*0000                        LD      INDEX,^LOPMSG;          % Output error message
0154   5F00*0000                        CALL    LPUTLINE;               %
0158   5E08*01EC                        JP      LRESTORE;               % Return to monitor
015C                    %
015C                                    EJECT;
```

```
MACRO8000:         Version 2.0   9/05/80                      Page 13
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

015C                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
015C                    %       Privileged Instruction Trap service routine
015C                    %
015C   5C09 000E*0000   LPRIV:          LDM     LSVAREA,R0,15;          % Save user regs 0-14
0162   7D07                             LDCTL   R0,NSPOFF;              % Save user R15
0164   6F00*001E                        LD      LSVAREA(30),R0;         %
0168                    %
0168   2109*0000                        LD      INDEX,^LPRMSG;          % Output error message
016C   5F00*0000                        CALL    LPUTLINE;               %
0170                    %
0170   5E08*01EC                        JP      LRESTORE;               % Return to monitor
0174                    %
0174                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0174                    %       This is the System Call handler routine.....
0174                    %
0174                    %       System Calls are treated as Breakpoints, Program Exits or I/O
0174                    %       Requests. An identifier of '00' is a Software Breakpoint, '01'
0174                    %       transfers to the console output handler and '02' transfers to
0174                    %       console output.
0174                    %
0174                    %       All other ID's are treated as program exits and the identifier
0174                    %       is output as part of the exit message.
0174                    %
0174   5C09 000E*0000   LSYSCALL:       LDM     LSVAREA,R0,15;          % Save user regs 0-14
017A   7D07                             LDCTL   R0,NSPOFF;              % Save user R15
017C   6F00*001E                        LD      LSVAREA(30),R0;         %
0180   21F7                             LD      HEX,STACKP^;            % Load identifier
0182                    %
0182                    %       This routine checks for a valid Breakpoint on a System Call '0'
0182                    %
0182                            IF      HEXLOW EQ 0
0182   84FF EE0A                        THEN    BEGIN
0186   31F0 0004                                LD      R0,STACKP^(4);  % Pick up PC from stack
018A   6107*0002                                LD      HEX,LBPSAV(2);  % Get saved address
018E   8370                                     SUB     R0,HEX;         % Compare address to PC
```



```
MACR08000:        Version 2.0   9/05/80                       Page 14
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

0190                                          IF      R0 EQ 2         % If addr compares transfer
0190  0B00 0002 EE02                          THEN    JP LBRKP        %   to Breakpoint routine
0196  5E08*0000                                       END;
019A                    %
019A                    %       This routine handles a request for console output (SC 1)
019A                    %
019A                                  IF      HEXLOW EQ 1
019A  0A0F 0101 EE03                  THEN    BEGIN
01A0  5F00*0000                               CALL    LPUTLINE;       % Output to CRT then return
01A4                                          IRET                    %    to user program
01A4  7B00                                    END;
01A6                    %
01A6                    %       This routine handles a request for console input (SC 2)
01A6                    %
01A6                                  IF      HEXLOW EQ 2
01A6  0A0F 0202 EE03                  THEN    BEGIN
01AC  5F00*0000                               CALL    LGETLINE;       % Input from CRT then return
01B0                                          IRET                    %    to user program
01B0  7B00                                    END;
01B2                    %
01B2                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
01B2                    %       Program Exit routine...
01B2                    %
01B2  5F00*0000                       CALL    LHEXVERT;                       % Convert identifier
01B6  4D05*0000 0002                  LD      LINEBUFF,#0002;                 % Store length char
01BC  6F07*0002                       LD      LINEBUFF(2),HEX;                % Store ident in buffer
01C0  2109*0000                       LD      INDEX,^LEXITMSG;                % Set R9 to exit message
01C4  5F00*0000                       CALL    LPUTLINE;                       % Output message
01C8  2109*0001                       LD      INDEX,^LINEBUFF+1;              % Set R9 to output
01CC  5F00*0000                       CALL    LPUTLINE;                       % Output identifier
01D0  5E08*01EC                       JP      LRESTORE;                       % Return to monitor
01D4                    %
01D4                                  EJECT;
```



```
MACRO8000:         Version 2.0   9/05/80                        Page 15
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE

01D4                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
01D4                    %       Break Switch service routine     (NMI)
01D4                    %
01D4   5C09 000E*0000   LBREAK:         LDM     LSVAREA,R0,15;          % Save user regs 0-14
01DA   7D07                             LDCTL   R0,NSPOFF;              % Save user R15
01DC   6F00*001E                        LD      LSVAREA(30),R0;         %
01E0   2109*0000                        LD      INDEX,^LBRKMSG;         % Output break message
01E4   5F00*0000                        CALL    LPUTLINE;               %
01E8   5E08*01EC                        JP      LRESTORE;               % Return to monitor
01EC                    %
01EC                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
01EC                    %       This common routine saves the user program status from the
01EC                    %       system stack into the program status save area for all the
01EC                    %       service routines. The routine also cleans up the stack pointer
01EC                    %       to prevent stack overflow from interrupts or traps.
01EC                    %
01EC   35F0 0002        LRESTORE:       LDL     RR0,STACKP^(2);         % Pick up program ststus
01F0   5D00*0000                        LDL     LSAVPS,RR0;             %     and save it
01F4   A9F5                             INC     STACKP,6;               % Restore stack pointer
01F6   5E08*009E                        JP      LCCP;                   % Exit to monitor CCP
01FA                    %
01FA                                    END.
```

```
MACR08000:         Version 2.0   9/05/80                      Page 16
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE



ASSIGNED LABELS:

LALTER    0000  EXTERNAL
LBPSAV    0000  EXTERNAL
LBPSET    0000  EXTERNAL
LBREAK    01D4  LOCAL     CODE
LBRKMSG   0000  EXTERNAL
LBRKP     0000  EXTERNAL
LCCP      009E  GLOBAL    CODE
LCMDTBL   0000  EXTERNAL
LDELAY    0056  LOCAL     CODE
LDUMP     0000  EXTERNAL
LEXITMSG  0000  EXTERNAL
LFLAG     0000  EXTERNAL
LGETLINE  0000  EXTERNAL
LGO       0000  EXTERNAL
LHELP     0000  EXTERNAL
LHEXVERT  0000  EXTERNAL
LINEBUFF  0000  EXTERNAL
LINITMSG  0000  EXTERNAL
LMFILL    0000  EXTERNAL
LMONIN    00A6  LOCAL     CODE
LNPSA     0000  LOCAL     NPSA
LOAD      0000  EXTERNAL
LOPCODE   0144  LOCAL     CODE
LOPMSG    0000  EXTERNAL
LPC       0000  EXTERNAL
LPRIV     015C  LOCAL     CODE
LPRMSG    0000  EXTERNAL
LPROMPT   0000  EXTERNAL
LPUTLINE  0000  EXTERNAL
LREG      0000  EXTERNAL
LRESTORE  01EC  LOCAL     CODE
```

```
MACRO8000:          Version 2.0   9/05/80                    Page 17
MACZ  M3EXEC O L=B:M3EXEC.PRN W D M P
MONITOR3 EXECUTIVE MODULE


LSAVPS    0000  EXTERNAL
LSSP      0000  EXTERNAL
LSTACK    0000  EXTERNAL
LSTART    0000  GLOBAL     CODE
LSVAREA   0000  EXTERNAL
LSYSCALL  0174  LOCAL      CODE
LWHAT     0138  GLOBAL     CODE
LWHATMSG  0000  EXTERNAL
```

APPENDIX G
ASCII CHARACTER SET

Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
00	0	NUL	20	32	SP	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	´	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

APPENDIX H

Powers of 2 and 16

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

Powers of 2

$2^{0} = 16^{0}$

$2^{4} = 16^{1}$

$2^{8} = 16^{2}$

$2^{12} = 16^{3}$

$2^{16} = 16^{4}$

$2^{20} = 16^{5}$

$2^{24} = 16^{6}$

$2^{28} = 16^{7}$

$2^{32} = 16^{8}$

$2^{36} = 16^{9}$

$2^{40} = 16^{10}$

$2^{44} = 16^{11}$

$2^{48} = 16^{12}$

$2^{52} = 16^{13}$

$2^{56} = 16^{14}$

$2^{60} = 16^{15}$

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

Powers of 16

A

APPENDIX I

Hexadecimal and Decimal Integer Conversion Table

8		7		6		5		4		3		2		1	
Hex	**Decimal**	**Hex**	**Decimal**	**Hex**	**Decimal**	**Hex**	**Decimal**	**Hex**	**Decimal**	**Hex**	**Decimal**	**Hex**	**Decimal**	**Hex**	**Decimal**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	268,435,456	1	16,777,216	1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	536,870,912	2	33,554,432	2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	805,306,368	3	50,331,648	3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	1,073,741,824	4	67,108,864	4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	1,342,177,280	5	83,886,080	5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	1,610,612,736	6	100,663,296	6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	1,879,048,192	7	117,440,512	7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	2,147,483,648	8	134,217,728	8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	2,415,919,104	9	150,994,944	9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	2,684,354,560	A	167,772,160	A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	2,952,790,016	B	184,549,376	B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	3,221,225,472	C	201,326,592	C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	3,489,660,928	D	218,103,808	D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	3,758,096,384	E	234,881,024	E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	4,026,531,840	F	251,658,240	F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
8		7		6		5		4		3		2		1	

To Convert Hexadecimal to Decimal

1. Locate the column of decimal numbers corresponding to the left-most digit or letter of the hexadecimal: select from this column and record the number that corresponds to the position of the hexadecimal digit or letter.
2. Repeat step 1 for the units (second from the left) position.
3. Repeat step 1 for the units (third from the left) position.
4. Add the numbers selected from the table to form the decimal number.

To convert integer numbers greater than the capacity of the table, use the techniques below:

Hexadecimal to Decimal

Successive cumulative multiplication from left to right, adding units position.

Example: D34$_{16}$ = 3380$_{10}$

```
D =    13
     x 16
      208
3 =  +  3
      211
     x 16
     3376
4 =   + 4
     3380
```

Conversion of Hexadecimal Value

	D34
1. D	3328
2. 3	48
3. 4	4
4. Decimal	3380

To Convert Decimal to Hexadecimal

1. (a) Select from the table the highest decimal number that is equal to or less than the number to be converted.
   (b) Record the hexadecimal of the column containing the selected number.
   (c) Subtract the selected decimal from the number to be converted.
2. Using the remainder from step 1(c) repeat all of step 1 to develop the second position of the hexadecimal (and a remainder).
3. Using the remainder from step 2 repeat all of step 1 to develop the units position of the hexadecimal.
4. Combine terms to form the hexadecimal number.

Decimal to Hexadecimal

Divide and collect the remainder in reverse order.

Example: 3380$_{10}$ = D34$_{16}$

```
16 |3380    remainder
16 |211  ──► 4    ▲
16 |13   ──► 3    │
         ──► D    │
```

Example:

Conversion of Decimal Value

	3380
1. D	−3328
	52
2. 3	−48
	4
3. 4	−4
4. Hexadecimal	D34